[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 7/24

[12] 发明专利申请公开说明书

[21] 申请号 00818392.9

[43] 公开日 2003 年 6 月 11 日

[11] 公开号 CN 1423772A

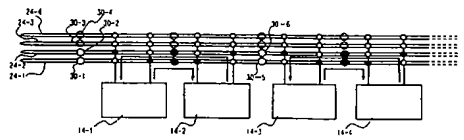
[22] 申请日 2000.11.21 [21] 申请号 00818392.9
[30] 优先权
[32] 1999.11.22 [33] JP [31] 330997/1999
[86] 国际申请 PCT/JP00/08209 2000.11.21
[87] 国际公布 WO01/38967 日 2001.5.31
[85] 进入国家阶段日期 2002.7.15
[71] 申请人 特博数据实验室公司
地址 日本神奈川
[72] 发明人 古庄晋二
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 李 强

权利要求书 6 页 说明书 39 页 附图 47 页

[54] 发明名称 信息处理系统

[57] 摘要

本发明的目的是以极高速度执行数据排序、汇编和联结。此目的系通过如下的分布式存储器型信息处理系统达到，其构成包括：一个 CPU 模块；每个都具有一个处理器和 RAM 核心的多个存储器模块；以及多组连接所述 CPU 和存储器模块和/或连接各个存储器模块的总线；其中的各个存储器模块的处理器执行由上述一个或多个存储器模块根据 CPU 发给各个存储器模块的处理器的指令进行管理的数组的处理，并且其中所述信息处理系统的特征在于所述存储器模块的构成包括：对组成数组的那些它自己管理的部分的元素执行排序并根据特定的顺序对所述元素重新排序、根据由其自身管理的所述部分在数组中所占据的位置将其 I/O、通过规定总线将所述排序后的元素与其顺序号一起发送到另一个存储器模块或通过规定总线从另一个存储器模块接收所述元素及其顺序号的排序装置；顺序号计算装置，此装置在接收到一个所述元素及顺序号时将其与由其自身管理的元素进行比较而计算出一个作为接收到的元素的顺序号的候选的虚拟顺序号，并且将其返回到所述另一个存储器模块；以及顺序确定装置，此装置在接收到所述虚拟顺序号时根据所述虚拟顺序号确定元素的顺序；结果所述数组的元素的顺序号是通过在发送所述元素及顺序号一方的展示存储器模块和在接收所述元素及顺序号并计算虚拟顺序号一方的存储器模块之间的通信来确定。



ISSN1008-4274

1.一种分布式存储器型信息处理系统，其构成包括：一个 CPU 模块；每个都具有一个处理器和 RAM 核心的多个存储器模块；以及多组连接所述 CPU 和存储器模块和/或连接各个存储器模块的总线；其中的各个存储器模块的处理器执行由上述一个或多个存储器模块根据 CPU 发给各个存储器模块的处理器的指令进行管理的数组的处理，并且其中

所述信息处理系统的特征在于所述存储器模块的构成包括：

对组成数组的那些它自己管理的部分的元素执行排序并根据特定的顺序对所述元素重新排序的排序装置，

根据由其自身管理的所述部分在数组中所占据的位置将其 I/O、通过规定总线将所述排序后的元素与其顺序号一起发送到另一个存储器模块或通过规定总线从另一个存储器模块接收所述元素及其顺序号的排序装置；

顺序号计算装置，此装置在接收到一个所述元素及顺序号时将其与由其自身管理的元素进行比较而计算出一个作为接收到的元素的顺序号的候选的虚拟顺序号，并且将其返回到所述另一个存储器模块；以及

顺序确定装置，此装置在接收到所述虚拟顺序号时根据所述虚拟顺序号确定元素的顺序；

结果所述数组的元素的顺序号是通过在发送所述元素及顺序号一方的展示存储器模块和在接收所述元素及顺序号并计算虚拟顺序号一方的存储器模块之间的通信来确定。

2.如权利要求 1 的信息处理系统，其中所述存储器模块的构成包括：

元素识别/发送装置，用来根据确定的顺序号识别要处理的元素并将其通过总线之一进行发送；

元素比较装置，用来对前一个待处理的元素与发送的元素进行比

较；以及

一个等值计数器，用来指示等同元素的计数并且其值在发送一个等同元素时递增；其中所述元素比较装置的构成使得当前一个待处理的元素被确定为与发送的元素不同时，此前一个待处理的元素与所述该元素相关的等值计数器的值相关联并且其一被送出，此外，并且

存储器模块之一接收前一个待处理的元素及如此发送的相关计数器的值，并且具有一个于其中使它们相关联并置于接收的顺序的数组。

3.如权利要求 2 的信息处理系统，其中所述存储器模块的构成包括：

一个指示非冗余顺序号的值-号计数器，其值在所述元素比较装置确定前一个待处理的元素与发送的元素不同时递增；以及

顺序号更新装置，其作用为，对发送的元素而言，在前一个待处理的元素与发送的元素等同时将值-号计数器的值设定为所述该非冗余元素的顺序号，但如果它们不同，将值-号计数器的递增值设定为所述该非冗余元素的顺序号。

4.一种通过利用分布式存储器型信息处理系统的数组排序方法，该系统的构成包括：一个 CPU 模块；每个都具有一个处理器和 RAM 核心的多个存储器模块；以及多组连接所述 CPU 和存储器模块和/或连接各个存储器模块的总线；其中的各个存储器模块的处理器执行由上述一个或多个存储器模块根据 CPU 发给各个存储器模块的处理器的指令进行管理的数组的处理，并且其中

所述排序方法的构成包括：

(a) 对组成数组的那些它自己管理的部分的元素执行排序的步骤，(b) 根据由其自身管理的所述部分在数组中所占据的位置在管理所述数组的位置的存储器模块中间确定发送元素及顺序号一方的展示存储器模块，以及在接收所述元素及顺序号一方的确定存储器模块的步骤，

(c) 在展示存储器模块中通过规定总线将排序后的元素与其顺序号一起发送到另一个存储器模块的步骤，

(d) 在确定存储器模块中通过规定总线从另一个存储器模块接收所述元素及其顺序号的步骤，

(e) 在所述确定存储器模块中，根据由所述确定存储器模块管理的元素的顺序号计算出指示一个接收到的元素的候选顺序号的虚拟顺序号，并且将所述虚拟顺序号返回到所述展示存储器模块的步骤，以及

(f) 在所述展示存储器模块中，当接收到所述虚拟顺序号时根据所述虚拟顺序号更新元素的顺序号的步骤，以及

(g) 在上述步骤(d)至(f)每一个步骤结束时，通过将属于藉助上述步骤(d)至(f)赋予一个规定顺序号的元素的展示存储器模块和确定存储器模块所组成的存储器模块组中的每一个分别取作展示存储器模块组和确定存储器模块组之一并且重复步骤(d)至(f)来更新每一个存储器模块组中的元素的顺序号，可确定数组中的每一个元素的顺序号.

5.如权利要求 4 的排序方法，其中所述步骤(e)的构成包括:

(e1) 根据指示要在接收到的元素的前方插入的元素数的前向插入数，要置于前方的元素的顺序号，以及接收到的顺序号计算虚拟顺序号的步骤。

6.如权利要求 4 或 5 的排序方法，其中所述步骤(f)的构成包括:

(f1) 将接收到的虚拟顺序号设置给在步骤(c)中发送的顺序号的步骤。

7.如权利要求 4 的排序方法，其构成还包括:

(h) 在展示存储器模块中，计算指示在所述存储器模块组中有多少个由组成所述展示存储器模块组的存储器模块管理的元素的冗余度，

其中所述步骤(c) 的构成包括:

(c1) 将排序后的元素及其顺序号和冗余度一起传送到另一个存储器模块以使等同的元素不会冗余传送的步骤，

其中所述步骤(e) 的构成包括:

(e1) 根据指示要在接收到的元素的前方插入的元素数的前向插入

数，要置于前方的元素的顺序号，以及接收到的顺序号和冗余度计算虚拟顺序号的步骤，并且

其中所述步骤(f)的构成包括:

(f2) 根据在步骤(c)中发送元素时虚拟顺序号和顺序号中间的差值确定与所述该元素等同的元素的顺序号的步骤。

8.如权利要求 4-7 任何一项中的排序方法，其中展示存储器模块是初始独立存储器模块，而接收模块也是独立存储器模块，并且

展示存储器模块组的构成包括 2^n 个存储器模块，其中的 n(n: 大于或等于 1 的整数)在步骤(d)至(f)每次结束时递增，同时确定存储器模块组的构成包括 2^n 个存储器模块。

9.一种汇编方法，其中一个数组是藉助于根据权利要求 4-6 之一中的方法排序，并且根据所述排序后数组生成一个新数组，在所述数组中元素按规定顺序排列，没有重复，并且其中所述汇编方法的构成包括:

(i) 在规定的存储器模块中根据顺序号发送待处理元素的步骤，

(j) 如前一个待处理的元素与发送的元素等同时将用来指示存在的等同元素的计数的值-号计数器的值递增，但如发送的元素与前一个待处理的元素不同时就将此前一个待处理的元素与所述该元素相关的等值计数器的值相关联并且将它们发送出去的步骤，

(k) 接收前一个待处理的元素及相关联的等值计数器的值，并且生成一个使它们相联系的新数组的步骤，以及

(l) 重复步骤步骤(i)至(j)以便使元素及其计数在新数组中互相联系。

10. 如权利要求 9 的汇编方法，其构成还包括:

(m) 在模块之一中监视在本周(j)中发送的元素及相联系的等值计数器的值的步骤，并且

其中步骤(k)藉助于所述模块之一执行。

11. 如权利要求 9 的汇编方法，其构成还包括:

(n) 在管理所述数组元素的存储器模块中，提供分别保持待处理

元素的顺序号和所述元素的计数的顺序号计数器和等值计数器，以及还提供临时存储前一个待处理元素的寄存器的步骤，

(o) 在管理具有所述该顺序号的元素的存储器模块中根据顺序号发送所述该元素的步骤，

(p) 在管理数组中的元素存储器模块中，比较接收的元素和寄存器的内容，并且如其等同，使计数递增，但如其不等同，就通过第二总线将寄存器的内容和计数器的值发送出去，并且之后更新寄存器的内容和计数器的值的步骤，

(q) 在存储器模块之一中，将所述寄存器的内容和所述计数器的值分别作为元素和所述元素的计数置于数组中的步骤。

12. 如权利要求 11 的汇编方法，其中步骤(n)的构成还包括：

(n1) 为待处理元素提供存储非冗余顺序号的值计数器的步骤，

并且所述步骤(p)的构成包括：

(p1) 比较接收的元素和寄存器的内容，并且如其等同，就将值-号计数器的值作为所述待处理元素的顺序号赋予，但如其不等同，使值-号计数器递增，并且将递增的值-号计数器的值作为所述待处理元素的顺序号赋予的步骤。

13. 一种联结数组的方法，系利用根据权利要求 4-8 之一中的排序方法和根据权利要求 9-12 之一中的汇编方法做到多个数组的联结，其中所述联结方法的构成包括：

(r) 通过对这些数组中的每一个元素赋予顺序号归并多个数组并执行所述排序方法的处理的步骤，以及

(s) 根据所述归并后的数组内的元素将其顺序号执行所述汇编方法的处理，从而生成没有冗余元素存在的新数组的步骤。

14. 一种联结数组的方法，系利用采用分布式存储器型信息处理系统的利用根据权利要求 4-8 之一中的排序方法和根据权利要求 9-12 之一中的汇编方法做到多个数组的联结，其中所述系统的构成包括：一个 CPU 模块；每个都具有一个处理器和 RAM 核心；以及多组连接所述 CPU 和存储器模块和/或连接各个存储器模块的总线；其中的各

个存储器模块的处理器执行由上述一个或多个存储器模块根据 CPU 发给各个存储器模块的处理器的指令进行管理的数组的处理，

所述存储器模块的构成包括一个其中的指示一个数值表的指针值置于与记录号相对应的位置以便指定数值表指定规定元素的指针数组，其中的数值表是一个根据记录号存储元素的数组，

并且上述联结方法的构成包括：

(r1) 通过对这些数组中的元素赋予顺序号归并多个数值表并执行所述排序方法处理的步骤，以及

(t) 根据所述归并后的数值表内的元素及其顺序号执行所述汇编方法，从而生成一个没有冗余元素存在的新数组并且还将所述元素的顺序号更新为没有冗余元素存在的场合的所述元素的顺序号的处理步骤，并且

(u) 将没有冗余元素存在的场合的所述元素的顺序号所组成的所述数组设置为用来指示新数值表的新指针数组。

信息处理系统

技术领域

本发明涉及分布式存储器型信息处理装置，特别涉及可以以极高速度执行数据排序、汇编和联结的信息处理装置。

背景技术

既然计算机已经进入整个社会的很多方面，并且因特网和其他网络已经普及，数据正在大规模地积累。这样大规模地处理数据要求巨大的计算能力，于是企图引入并行处理是很自然的。

当前，并行处理体系结构分为“共享存储器”型和“分布式存储器”型。前者(“共享存储”型)是多个处理器共享单个巨大存储器空间的体系结构。在此体系结构中，处理器组和共享存储器之间的通信量是一个瓶颈，所以不容易构建使用 100 个以上处理器的实际系统。因此，在计算 10 亿个浮点数的平方根时，比如，处理的执行速度不会快于单个 CPU 的速度的 100 倍。从经验得知，上限是大约 30 倍。

在后者(“分布式存储器”型)的情况下，每个处理器具有其自己的局部存储器，并且两者连接而构建一个系统。采用这一体系结构时，可以设计一个整合甚至数百个至数万个处理器的硬件系统。因此，在在计算上述 10 亿个浮点数的平方根时，比如，处理的执行速度可比单个 CPU 的速度快上数百倍至数万倍。

据说是对于由数目达到数百或更多的大量处理器实现的并行处理的潜在需求是巨大的，但是，如上所述，如果要利用现有的现实的硬件技术来实现，除了采用“分布式存储器”型之外是很难设计的。

在分布式存储器体系结构中，附加于单个处理器上的存储器的容量很小，因此在并行处理的主要对象之一的大规模数据(通常是数组)的存储和处理中必须将此数据在多个处理器和附加于每个处理器的存

储器中间划分。

然而，当数组在多个处理器和附加于每个处理器的存储器中间划分时，防止数据在总线上的冲突的总线主控变得很困难，因此如各个处理器不能并行工作，那就存在一个问题，就是无法提高处理器的使用效率而不可能增加处理速度。为此，本发明要达到的各个目的如下所述。

(1) 数据在总线上的冲突在算法上不会发生，因此总线主控就不需要了；从而通过充分使用总线的带宽可以增加处理速度。

(2) 通过将多个装设有处理器的存储器模块与处理器(最好是多个处理器)相结合可以进行并行处理，并且使各个存储器模块可以得到有效的利用，而处理可独立地分配给每个存储器模块中的处理器，从而可通过有效的使用存储器模块而进一步增加处理速度。

(3) 如待排序的数据的量是 N，则只需要 O(N)的数据量。(在普通排序时必需 O(N*N)的数据量或在最坏的情况下为 O(N*Log(N))的数据量。)

(4) 处理时间稳定，即使是在最坏的情况下也是，预测 C 处理速度可以保证。

就是说，本发明的目的是提供一种可以以极高速度和稳定的处理时间执行数组排序的信息处理装置。

发明概述

本发明的一个目的可以通过如下的分布式存储器型信息处理系统达到，其构成包括：一个 CPU 模块；每个都具有一个处理器和 RAM 核心的多个存储器模块；以及多组连接所述 CPU 和存储器模块和/或连接各个存储器模块的总线；其中的各个存储器模块的处理器执行由上述一个或多个存储器模块根据 CPU 发给各个存储器模块的处理器的指令进行管理的数组的处理，并且其中所述信息处理系统的特征在于所述存储器模块的构成包括：对组成数组的那些它自己管理的部分的元素执行排序并根据特定的顺序对所述元素重新排序、根据由其自

身管理的所述部分在数组中所占据的位置将其 I/O、通过规定总线将所述排序后的元素与其顺序号一起发送到另一个存储器模块或通过规定总线从另一个存储器模块接收所述元素及其顺序号的排序装置；顺序号计算装置，此装置在接收到一个所述元素及顺序号时将其与由其自身管理的元素进行比较而计算出一个作为接收到的元素的顺序号的候选的虚拟顺序号，并且将其返回到所述另一个存储器模块；以及顺序确定装置，此装置在接收到所述虚拟顺序号时根据所述虚拟顺序号确定元素的顺序；结果所述数组的元素的顺序号是通过在发送所述元素及顺序号一方的展示存储器模块和在接收所述元素及顺序号并计算虚拟顺序号一方的存储器模块之间的通信来确定。

藉助本发明，由展示存储器模块对元素和顺序号的展示是通过总线执行，而虚拟顺序号是利用确定存储器模块计算，并且这些虚拟顺序号是通过另外一个总线赋予展示存储器模块。因此，在展示存储器模块和确定存储器模块中排序可以并行进行，并且总线冲突可以避免。

在实施本发明的具体方式中，所述存储器模块的构成包括：元素识别/发送装置，用来根据确定的顺序号识别要处理的元素并将其通过总线之一进行发送；元素比较装置，用来对前一个待处理的元素与发送的元素进行比较；以及一个等值计数器，用来指示等同元素的计数并且其值在发送一个等同元素时递增；其中所述元素比较装置的构成使得当前一个待处理的元素被确定为与发送的元素不同时，此前一个待处理的元素与所述该元素相关的等值计数器的值相关联并且其一被送出，此外，并且存储器模块之一接收前一个待处理的元素及如此发送的相关计数器的值，并且具有一个于其中使它们相关联并置于接收的顺序的数组。

藉助于此具体实施方式，在存储器模块之一中，元素及其冗余度是以规定顺序接收，因而就可以生成一个非冗余元素数组和各个元素的计数。就是说，从而就可以很容易地确定一个非冗余元素列表，并且原始数组中的每个元素号可以很容易确定。

在本发明的另一个具体实施方式中，所述存储器模块的构成包括：

一个指示非冗余顺序号的值-号计数器，其值在所述元素比较装置确定前一个待处理的元素与发送的元素不同时递增；以及顺序号更新装置，其作用为，对发送的元素而言，在前一个待处理的元素与发送的元素等同时将值-号计数器的值设定为所述该非冗余元素的顺序号，但如果它们不同，将值-号计数器的递增值设定为所述该非冗余元素的顺序号。

藉助于此具体实施方式，可以将应用于数组元素的顺序号变换为其中的元素冗余度被消除的状态中的 1。

此外，本发明的一个目的可以通过利用分布式存储器型信息处理系统的数组排序方法来达到，此系统的构成包括：一个 CPU 模块；每个都具有一个处理器和 RAM 核心的多个存储器模块；以及多组连接所述 CPU 和存储器模块和/或连接各个存储器模块的总线；其中的各个存储器模块的处理器执行由上述一个或多个存储器模块根据 CPU 发给各个存储器模块的处理器的指令进行管理的数组的处理，并且其中所述排序方法的构成包括：

(a) 对组成数组的那些它自己管理的部分的元素执行排序的步骤，(b) 根据由其自身管理的所述部分在数组中所占据的位置在管理所述数组的位置的存储器模块中间确定发送元素及顺序号一方的展示存储器模块，以及在接收所述元素及顺序号一方的确定存储器模块的步骤，

(c) 在展示存储器模块中通过规定总线将排序后的元素与其顺序号一起发送到另一个存储器模块的步骤，

(d) 在确定存储器模块中通过规定总线从另一个存储器模块接收所述元素及其顺序号的步骤，

(e) 在所述确定存储器模块中，根据由所述确定存储器模块管理的元素的顺序号计算出指示一个接收到的元素的候选顺序号的虚拟顺序号，并且将所述虚拟顺序号返回到所述展示存储器模块的步骤，以及

(f) 在所述展示存储器模块中，当接收到所述虚拟顺序号时根据

所述虚拟顺序号更新元素的顺序号的步骤，以及

(g) 在上述步骤(d)至(f)每一个步骤结束时，通过将属于藉助上述步骤(d)至(f)赋予一个规定顺序号的元素的展示存储器模块和确定存储器模块所组成的存储器模块组中的每一个分别取作展示存储器模块组和确定存储器模块组之一并且重复步骤(d)至(f)来更新每一个存储器模块组中的元素的顺序号，可确定数组中的每一个元素的顺序号。

藉助本发明，展示存储器模块中的计算，展示存储器模块中的元素和顺序号的发送，确定存储器模块组中的计算和展示存储器模块中的虚拟顺序号的发送可以并行进行，并且总线冲突可以避免。就是说，可以获得极高的速度的排序(对数组中的元素赋予顺序号)。此外，存储器中的数据量可保持为只是O(N)。

在上述实施本发明的具体方式中，步骤(e)的构成包括:

(e1) 根据指示要在接收到的元素的前方插入的元素数的前向插入数，要置于前方的元素的顺序号，以及接收到的顺序号计算虚拟顺序号的步骤。

在上述本发明的另一具体实施方式中，步骤(f)的构成包括:

(f1) 将接收到的虚拟顺序号设置给在步骤(c)中发送的顺序号的步骤。

实施本发明的具体方式的构成还包括:

(h) 在展示存储器模块中，计算指示在所述存储器模块组中有多少个由组成所述展示存储器模块组的存储器模块管理的元素的冗余度，

其中所述步骤(c) 的构成包括:

(c1) 将排序后的元素及其顺序号和冗余度一起传送到另一个存储器模块以使等同的元素不会冗余传送的步骤，

其中所述步骤(e) 的构成包括:

(e1) 根据指示要在接收到的元素的前方插入的元素数的前向插入数，要置于前方的元素的顺序号，以及接收到的顺序号和冗余度计算虚拟顺序号的步骤，并且

其中所述步骤(f)的构成包括:

(f2) 根据在步骤(c)中发送元素时虚拟顺序号和顺序号中间的差值确定与所述该元素等同的元素的顺序号的步骤。

藉助于本发明，无需展示存储器模块多次发送同一元素。此外，当计算出某一元素的冗余度时，可以将所述该元素的顺序号和冗余度发送给确定存储器模块，并且可在确定存储器模块中对所述该元素执行虚拟顺序号的计算。就是说，这可以防止存储器模块利用效率的降低。

在另一具体实施方式中，展示存储器模块是初始独立存储器模块，而接收模块也是独立存储器模块，并且展示存储器模块组的构成包括2^n个存储器模块，其中的n(n: 大于或等于1的整数)在步骤(d)至(f)每次结束时递增，同时确定存储器模块组的构成包括2^n个存储器模块。

如上所述，在理想情况下当使用2^n个存储器模块时可做到排序。

此外，本发明的另一具体实施方式是一种汇编方法，其中一个数组是藉助于上述排序方法排序，并且根据所述排序后数组生成一个新数组，在所述数组中元素按规定顺序排列，没有重复，并且其中所述汇编方法的构成包括:

(i) 在规定的存储器模块中根据顺序号发送待处理元素的步骤，

(j) 如前一个待处理的元素与发送的元素等同时将用来指示存在的等同元素的计数的值-号计数器的值递增，但如发送的元素与前一个待处理的元素不同时就将此前一个待处理的元素与所述该元素相关的等值计数器的值相关联并且将它们发送出去的步骤，

(k) 接收前一个待处理的元素及相关联的等值计数器的值，并且生成一个使它们相联系的新数组的步骤，以及

(l) 重复步骤步骤(i)至(j)以便使元素及其计数在新数组中互相联系。

此外，上述汇编方法的构成还可包括:

(m) 在模块之一中监视在本周(j)中发送的元素及相联系的等值计数器的值的步骤，并且

其中步骤(k)藉助于所述模块之一执行。

此外，上述汇编方法的构成还可包括:

(n) 在管理所述数组元素的存储器模块中，提供分别保持待处理元素的顺序号和所述元素的计数的顺序号计数器和等值计数器，以及还提供临时存储前一个待处理元素的寄存器的步骤，

(o) 在管理具有所述该顺序号的元素的存储器模块中根据顺序号发送所述该元素的步骤，

(p) 在管理数组中的元素存储器模块中，比较接收的元素和寄存器的内容，并且如其等同，使计数递增，但如其不等同，就通过第二总线将寄存器的内容和计数器的值发送出去，并且之后更新寄存器的内容和计数器的值的步骤，

(q) 在存储器模块之一中，将所述寄存器的内容和所述计数器的值分别作为元素和所述元素的计数置于数组中的步骤。

此外，步骤(n)的构成还包括:

(n1) 为待处理元素提供存储非冗余顺序号的值计数器的步骤，

并且所述步骤(p)的构成最好还包括:

(p1) 比较接收的元素和寄存器的内容，并且如其等同，就将值-号计数器的值作为所述待处理元素的顺序号赋予，但如其不等同，使值-号计数器递增，并且将递增的值-号计数器的值作为所述待处理元素的顺序号赋予的步骤。

此外，在上述本发明的另一具体实施方式中，联结数组的方法可利用上述排序方法和上述汇编方法做到多个数组的联结，其中所述联结方法的构成包括:

(r) 通过对这些数组中的每一个元素赋予顺序号归并多个数组并执行所述排序方法的处理的步骤，以及

(s) 根据所述归并后的数组内的元素将其顺序号执行所述汇编方法的处理，从而生成没有冗余元素存在的新数组的步骤。

就是说，通过在所要求的数组得到归并的状态下执行根据本发明的排序和汇编，可以得到消除元素冗余的联结数组。

在上另外又一个具体实施方式中，联结数组的方法可利用采用分布式存储器型信息处理系统的上述排序方法和上述汇编方法做到多个数组的联结，其中所述系统的构成包括：一个 CPU 模块；每个都具有一个处理器和 RAM 核心；以及多组连接所述 CPU 和存储器模块和/或连接各个存储器模块的总线；其中的各个存储器模块的处理器执行由上述一个或多个存储器模块根据 CPU 发给各个存储器模块的处理器的指令进行管理的数组的处理，

所述存储器模块的构成包括一个其中的指示一个数值表的指针值置于与记录号相对应的位置以便指定数值表指定规定元素的指针数组，其中的数值表是一个根据记录号存储元素的数组，

并且上述联结方法的构成包括：

(r1) 通过对这些数组中的元素赋予顺序号归并多个数值表并执行所述排序方法处理的步骤，以及

(t) 根据所述归并后的数值表内的元素及其顺序号执行所述汇编方法，从而生成一个没有冗余元素存在的新数组并且还将所述元素的顺序号更新为没有冗余元素存在的场合的所述元素的顺序号的处理步骤，并且

(u) 将没有冗余元素存在的场合的所述元素的顺序号所组成的所述数组设置为用来指示新数值表的新指针数组。

附图简述

本发明的各个目的参考附图及具体实施方式将一清二楚。其中：

图 1 为示出根据本发明的一个具体实施方式的计算机系统的配置的框图。

图 2 为示出根据此具体实施方式的存储器模块的示意框图。

图 3 为用来描述根据本具体实施方式的存储器模块中间的流水线处理的示图。

图 4A 至 4C 为用来描述根据本具体实施方式的多空间存储器下的存储器模块 14 的结构的示图。

图 5A 至 5C 为用来描述根据本具体实施方式的存储器模块的访问的的示图。

图 6A 至 6C 为示出对一个数组执行根据本具体实施方式 1 的排序的一个示例的示图。

图 7 为示出根据本具体实施方式 1 的排序进程的流程图。

图 8 为示出在执行根据具体实施方式 1 的排序时存储器模块之间的连接的框图。

图 9 是示意图，显示了图 8 所示的存储器模块之间的连接。

图 10A 至 10D 为示出在根据本具体实施方式 1 的排序中在一个数组内的元素的编号的示图。

图 11A 至 11D 为示出在根据本具体实施方式 1 的排序中在一个数组内的元素的顺序编号的示图。

图 12A 至 12C 为示出在根据本具体实施方式 1 的排序中在一个数组内的元素的编号的示图。

图 13A 至 13C 为示出在根据本具体实施方式 1 的存储器模块对之间的顺序编号的编号的流程图。

图 14A 和 14B 为示出属于示于图 8 中的存储器模块的两个存储器模块组之间的连接示例的示图。

图 15A 和 15B 为示出示于图 14 的连接的示例的示意图。

图 16A 和 16B 为示出在根据本具体实施方式 1 的排序中在一个数组内的元素的顺序编号的示图。

图 17A 和 17B 为示出在根据本具体实施方式 1 的排序中在一个数组内的元素的顺序编号的示图。

图 18A 和 18B 为示出在根据本具体实施方式 1 的排序中在一个数组内的元素的顺序编号的示图。

图 19A 和 19B 为示出在根据本具体实施方式 1 的排序中在一个数组内的元素的顺序编号的示图。

图 20 为用来描述在根据本具体实施方式 1 的排序中存储器模块的组合的示图。

图 21 为示出在按照作为根据本具体实施方式 1 的排序的结果而取

得的顺序号生成一个新数组的场合中的存储器模块的连接的另一示例的示图。

图22为示出在按照作为根据本具体实施方式1的排序的结果而取得的顺序号生成一个新数组的场合中的存储器模块的连接的另一示例的示图。

图23为示出在根据本具体实施方式2的排序中存储器模块的连接的示例的示意图。

图24为用来描述在根据本具体实施方式2的存储器模块组中计算冗余度的进程的示图。

图25为用来描述在根据本具体实施方式2的存储器模块组中计算冗余度的进程的示图。

图26为示出在根据本具体实施方式2的存储器模块组中计算冗余度的进程的流程图。

图25为用来描述在根据本具体实施方式2的存储器模块组中计算冗余度的进程的示图。

图28A和28B为用来描述在根据本具体实施方式2的存储器模块组中计算冗余度的进程的示图。

图29A和29B为用来描述在根据本具体实施方式2的存储器模块组中计算冗余度的进程的示图。

图30A和30B为用来描述在根据本具体实施方式2的存储器模块组中计算冗余度的进程的示图。

图31A至31C为示出在排序同时省略发送冗余元素的的流程图。

图32为示出在按照作为根据实施本发明的具体方式3的汇编中的存储器模块的连接的一个示例的示图。

图33为示出根据实施本发明的具体方式3的汇编的流程图。

图34A和34B为用来描述在根据实施本发明的具体方式3的存储器模块组中的汇编的示图。

图35A和35B为用来描述在根据实施本发明的具体方式3的存储器模块组中的汇编的示图。

图 36A 和 36B 为用来描述在根据实施本发明的具体方式 3 的存储器模块组中的汇编的示图。

图 37A 和 37B 为用来描述在根据实施本发明的具体方式 3 的存储器模块组中的汇编的示图。

图 38 为用来描述在根据实施本发明的具体方式 3 的存储器模块组中的汇编的示图。

图 39A 和 39B 为用来描述在根据实施本发明的具体方式 3 的存储器模块组中的汇编的示图。

图 40 为示出根据实施本发明的具体方式 4 的联结的流程图。

图 41A 和 41B 为用来描述在根据实施本发明的具体方式 4 的存储器模块组中的联结的示图。

图 42A 至 42C 为用来描述在根据实施本发明的具体方式 4 的存储器模块组中的联结的示图。

图 43A 和 43B 为用来描述在根据的实施本发明的具体方式 4 的存储器模块组中的联结的示图。

图 44A 和 44B 为用来描述在根据实施本发明的具体方式 4 的存储器模块组中的联结的示图。

图 45 为示出本发明的另一实例中的存储器模块组中的联结的示意图。

实施发明的具体方式

[硬件配置]

下面是参考附图对本发明的具体实施方式的描述。图 1 为示出根据本发明的一个具体实施方式的计算机系统的配置的框图。如图 1 所示，计算机系统 10 的构成包括根据单个指令执行并行运算的 CPU 模块 12；存储并行运算所需的各个类型的数据的存储器模块 14-1、14-2、14-3；存储所需程序和数据的硬盘驱动器 16；键盘、鼠标或其他输入装置 18；由 CRT 等等组成的显示器 20 以及以各种格式存储数据等等

的遗留存储器 22。此外，在总线 24-1，24-2，...，上，在与各个存储器模块 14 的接触点处设置开关 28-1，28-2，28-3，...等等，所以选择的电路元件可以交换信息。此外，在 CPU 模块 12 和存储器模块 14-1 中间设置有开关 30-1，30-2，...用于总线连接和相邻存储器模块中间的连接。此外，在存储器模块输入端到总线的接触点和存储器模块输出端到总线的接触点中间也可以设置一个开关(见符号 29)。在图 1 中，上述开关以虚线圆圈表示。

另外，存储器模块 14 最好是不但具有单输入端和单输出端，而且还具有一个或多个其他终端(I/O 终端等等)。比如，在后面将要介绍的具体实施方式 2 或 3 中，处理是利用 3 个以上的终端的 I/O 实现的。

在 CPU 模块 12 和存储器模块 14 中间设置多个总线 24-1，24-2，24-3，24-4，...。因此，数据等等可藉助上述总线在存储器模块之间交换。此外，在 CPU 12 和存储器模块 14 中间设置有一根信号控制线 25，于是由 CPU 12 发出的指令可传输到所有的存储器模块 14。

另外，在 CPU 12 和各个其他构件(例如，硬盘驱动器 16，输入装置 18 等等)中间设置有本地总线 26，于是数据等等也可在其中间交换。CPU 12 读取存储于与本地总线 26 相连接的硬盘驱动器 16 或 RAM 或其他存储装置(未示出)上的程序，并按照该程序执行发送指令到存储器模块 14 和其他数据交换，另外还控制开关 28，30 等等。此外，根据此程序，CPU 12 接受存储于遗留存储器 22 中的各种格式的数据，将此格式数据变换为可利用由 CPU 12、存储器模块 14 和总线 24 构成的系统处理的数据系列(数组)并将其存储于不同的存储器模块 14 中。

图 2 为示出根据此具体实施方式的存储器模块的示意框图。如图 2 所示，存储器模块 14 的构成包括一个接受由 CPU 模块 12 发出的时钟或其他同步信号的时钟缓存 32；一个此处数据的 RAM 核心 34；一个处理器(MPU)36，此处理器识别空间 ID 和数据的元素号(见后述)并在接到 CPU 12 的指令等时根据空间 ID 和数据的元素号将数据写入 RAM 核心 34 或从 RAM 核心 34 读出数据；以及一个 I/O 单元 38，此 I/O 单元从总线之一接受数据并将其供给 RAM 核心 34，和/或从 RAM

核心 34 送出并在总线之一上将其送出。在此具体实施方式中，存储器模块 14 可通过信号控制线 25 接受来自 CPU 的指令，对这些指令做出响应而执行从 RAM 核心 34 读出数据，将数据写入 RAM 核心 34 或对数据执行其他规定的处理。此外，对 RAM 核心 34 的数据访问和通过 I/O 单元的数据输入和输出是根据时钟缓存 32 发出的时钟或其他同步信号执行的。上述存储器模块 14 的处理器(MPU)36 最好是由多个处理单元组成并且能够执行多个进程。

从图 1 和图 2 可知，在本发明中，计算机系统 10 可看作是一个共享存储器型系统。此外，数据输出到总线和从总线输入数据等等是根据规定的同步信号执行。因此，此计算机系统 10 可认为是采用 SIMD(单指令流多数据流)系统的形式。

具有这种配置的计算机系统 10 基本上配置有多空间存储器，存储器模块和在日本专利申请 No. H11-263793 中记载的可重构总线。下面对这些予以简要描述。

(1) 在此说明书中，“多空间存储器”指的是存储器空间分配为根据空间 ID 和地址进行访问。因此，即使是一个数据序列在很多处理器之间分割，每个处理器都能够可靠地将其分离和识别。

在通常的存储器空间中，即使是对每个进程分配单独区域，对每个变量序列(数组，结构等等)并不执行存储器空间分配。因此，下面将把通常的存储器空间描述为“单存储器空间”。在单存储器空间系统中，数据访问只利用地址进行，所以不可能分离或识别相关数据序列。由于这一原因，即使是并行处理实际上可能，也有很多场合对此无法确定。此外，当将一个新的数据序列存储于某个单存储器空间时，必须执行无用存储单元收集(废料收集)以保证用来存储所述该数据序列的地点安全可靠。

与此相反，在本发明中，在存储器空间引入空间 ID，从而可将同一 ID 应用于数据序列。此外，存储器模块 14 中的每一个都可以识别保持于 RAM 核心 34 内的数据空间 ID，并从而每个存储器模块 14 都能够通过检查当前正在访问的数据空间 ID 来确定它是否需要进行运

算。此外，因为每个存储器模块14可保持与空间ID相关联的数据序列的全部或某些，所以就可能存储在多个存储器模块14之间分割的某一个数据序列，并且从而无用存储单元收集(废料收集)就不需要了。

(2) 存储器模块

此外，在本发明中，存储器模块14中的每一个可识别它自己保持的数据序列的单个元素号的处理器36。因此，在接到来自CPU 12的指令时，处理器36可以确定根据指令要访问的数据是否保持于RAM核心34内并从而确定是否需要访问。另外，存储器模块14中的每一个都可以从存储于其固有的RAM核心34中的数组元素中的下标范围确定在SIMD下的指令隐含处理中的负担范围。

存储器模块14可以根据来自CPU 12的指令对待处理的元素的次序重排，并且对存储于其固有的RAM核心34中的元素排序。

(3) 可重构总线

采用本发明，CPU 12可以有选择地接通/关断开关28-1，28-2，...和开关30-1，30-2，...，并且这样就可指定将要与其进行数据交换的存储器模块14，从而做到流水线处理。比如，如图3所示，如从某一个存储器模块14-i输出的数据需要交给另外一个存储器模块14-j，并且从此另外一个存储器模块14-j输出的数据要传送到再另外一个存储器模块14-k，那末CPU 12就将各个开关设置成为总线24-m分配给存储器模块14-i和存储器模块14-i，而总线24-n分配给存储器模块14-j和存储器模块14-k。

另外，此流水线处理不仅可在单存储器模块之间连接的场合做到，而且可以通过多个存储器模块序列(存储器模块组)之间的连接来做到。取决于要进行的处理，各个存储器模块可重新连接以便可以在每个连接路径上以规定的次序进行规定类型数据的单向连续传送，从而对通信进行调度以使接近100%的总线容量可以得到利用。于是，就可以解决作为分布式存储器型并行处理系统的最大问题的处理器间通

信性能低下的问题。

[多空间存储器]

下面我们将更详细描述在根据采用多空间存储器的本发明的计算机系统中按照指令对各个存储器模块的存储器管理和存储器访问。

图 4A 至 4C 为用来描述根据本具体实施方式的多空间存储器下的存储器模块 14 的结构的示图。如图 4A 所示，在存储器模块 14 内的 RAM 核心 34 中设置有空间 ID 控制表。于是，存储器模块 14 的处理器 36 可确定由其自身保持的空间 ID 和其他必要的信息。

如图 4B 所示，空间 ID 控制表包含其保持的每个数据组的空间 ID，CPU 控制下的逻辑起始地址，分配给数据组的区域的大小，RAM 核心 34 内的物理起始地址，具有所述该空间 ID 的数据序列的总容量以及控制访问的访问控制标记。访问控制标记可设定为以下三种状态之一：只读(R)，只写(W)或允许读/写(RW)。

当给定一个具有空间 ID 的数据组时，存储器模块 14 的处理器 36 在 RAM 核心 34 中找到一个或多个可存储所述该数据组的区域并将数据组存储于所述区域，按原样或分割为两个或多个部分。此时，RAM 核心中实际存储数据的逻辑起始地址和分配区域的大小与给定的空间 ID，逻辑起始地址，总容量和访问控制标记一起存储于空间 ID 控制表中。图 4 处理器为示出根据图 4 的空间 ID 控制表存储于 RAM 核心 34 内的数据的示图。

[存储器访问简议]

下面对具有这种构成的存储器模块 14 的访问予以描述。如图 5A 至 5C 所示，CPU 12 首先通过信号控制线 25 向使用的存储器模块 14 传送空间 ID，逻辑起始地址和所要求的指令(比如读写数据)。对此的响应是存储器模块 14 中的每一个利用在处理器 36 中设置的空间比较器 52 比较此空间 ID 和保持于其固有的空间 ID 控制表中的空间 ID 并

确定其自身保持的是否一样，另外，地址比较器 54 对逻辑地址执行同样类型的确定。其后，假如确定有待指令处理的数据是保持于其固有的 RAM 核心 34 内，存储器模块 14 的处理器 36 利用地址计算器 56 检查空间 ID 控制表以便计算 RAM 核心 34 内的物理地址和识别有待处理的数据。

一旦数据以这种方式被识别，处理器 36 就执行与 CPU 12 发出的指令相应的处理(比如读写数据)，并且，需要时，将此数据传送到 CPU 12(见图 5 处理器)。

[排序(具体实施方式 1)]

下面对藉助具有这种构成的计算机系统 10 的排序予以描述。注意在下面描述中的每个存储器模块都带有一个处理器并且因而称为处理器存储模块(处理器存储模块)。

为易于理解，考虑如图 6A 至 6C 所述的情况，其中四个 PMM 中的每一个都保持两个元素(姓)。如图 6A 所示，某一个 PMM(第一个 PMM14-1)保持元素下标(记录号)为 “0”的姓 “Smith”和元素下标为 “1”的姓 “Carter”。第二个 PMM14-2 保持元素下标为 “2”的姓 “Clinton”和元素下标为 “3”的姓 “Johnson”。与此类似，第第三个 PMM14-3 和第四个 PMM14-4 保持与示于图 6A 的下标相对应的的姓。同一空间 ID 应用于由这些元素组成的数组，并且每个 PMM 的处理器 36 都利用空间 ID 控制表来管理在其固有的 RAM 核心 34 中的元素的下标(记录号)，还有它们实际存储的物理地址等等。

比如，考虑由 CPU 12 通过信号控制线 25 向各个 PMM14-1 至 14-4 发出对具有此空间 ID 的数组进行排序的指令。图 7 为示出根据本具体实施方式的排序进程的流程图。如图 7 所示，当 CPU 12 发出指令(比如，指令 “对具有某一空间 ID 的数组内的元素排序”)(步骤 700)时，响应此指令，在每个 PMM 中，每个 PM 的处理器 36 都接收到通过信号控制线 25 给出的指令并解释其内容(步骤 701)，检查指令内的 “空间 ID”(步骤 702)，并确定它是否属于由其固有的 RAM 核心 34

保持的数据的空间 ID(步骤 703)。如步骤 703 的结果为 “否”，则处理结束。另一方面，如结果为 “是”，则处理器 36 执行所要求的各种检验，如检验属于所述该空间 ID 的数据组是处于允许写入状态(步骤 704)。如检验出现出错结果(步骤 705 中的 “是”)，则处理器 36 通过信号控制线 25 向 CPU 12 报告出错。另一方面，如没有出错，则处理器 36 执行下面描述的排序处理的主体(步骤 707 及其以下步骤)。

首先，PMM 14-1 至 14-4 中的每一个都对其固有的元素执行排序(步骤 707)。此排序实际上伴随有元素在各个 PMM 内的位置。更具体说，处理器 36 利用 Quicksort 或其他熟知的排序技术来对保持于其固有的 RAM 核心 34 中的元素排序。图 6B 为示出每个 PMM 的数组内的元素的排序状态的示图。注意，如图 6D 所示，每个元素的下标(记录号)的位置也改变。

其次，每个存储器模块 14 的处理器 36 分配一个区域(顺序号区域)用来置放顺序号并给出每个顺序号的初始值(步骤 708)。图 6C 为示出其中顺序号的初始值已经给出的每个 PMM 的状态的示图。这样，就在每个模块内在排序后的元素中间赋予顺序号。

之后，在邻接对之间执行顺序号的归并和赋予(步骤 709)。在步骤 709 中，CPU 12 首先控制总线上的开关 28 和 30 使得在排序中涉及的 PMM 之间对某一对的一侧的输入连接到另一侧的输出，而那一侧的输出连接到另一侧的输入。上述的对最好是两个邻接的 PMM 或者，如果不是邻接的，是两个处于附近位置的 PMM。比如，在图 1 中，如在排序中涉及 PMM 14-1 至 14-4，PMM 14-1 和 PMM 14-2 和 PMM 14-3 和 PMM 14-4 最好是配对。比如，如图 8 所示，CPU 12 可控制开关 28 以使 PMM 14-1 的输出和 PMM 14-2 的输入连接到总线 24-1，PMM 14-1 的输入和 PMM 14-2 的输出连接到总线 24-2，并且制开关 28 以使 PMM 14-3 的输出和 PMM 14-4 的输入连接到总线 24-1，PMM 14-3 的输入和 PMM 14-4 的输出连接到总线 24-2。另外，CPU 12 还关断置于 PMM 14-2 和 PMM 14-3 之间的总线 24-1 和 24-2 上的开关

30-5 和 30-6。在图 8 中，黑圈表示连续状态，而白圈表示不连续状态和不与 PMM 连接。此外，其余的一些遵循其他 PMM(未示出)的状态。注意在图 8 的示例中可以了解到通过关断开关 30-5 和 30-6 将总线 24-1 和 24-2 分割可以更有效地利用总线。

这样，如图 9 示意地示出，PMM 之间的连接由 CPU 12 确定。并且顺序号赋予的主体是在 PMM 对之间执行。

图 10A 至 12C 只示出 PMM 14-1 和 PMM 14-2 的处理，但 PMM 14-3 和 PMM 14-4 的处理也并行地执行。注意在处理中首先将数据给予另一个 PMM 的称为前 PMM，而接受数据的(另一个 PMM)称为后 PMM。前 PMM 也可称为展示 PMM，因为它展示元素或顺序号，而另一方面，后 PMM 也可称为确定 PMM，因为它确定展示的顺序号。PMM 对之中的哪一个都可以是前 PMM。在此示例中，为方便起见，将 PMM 14-1 作为前 PMM，而将 PMM 14-2 作为后 PMM。

首先，在前 PMM 中，指示处理位置的指针(以后称其为 “PUT 指针”)置于初始位置(排序数组部分的开始处或 “第 0”位置)。另一方面，在后 PMM 中，如下所述，从前 PMM 接受的元素和首先指示待比较位置等等的指针(以后称其为 “比较指针”)置于初始位置(排序数组部分的开始处或 “第 0”位置)。(见图 10A 和图 13A 及图 13B 的步骤 1301 和 1311)。在此具体实施方式中，在后 PMM 中使用的比较指针是形式(X，Y，Z)的结构数组。此处，X 表示待比较的开始位置(即还有待比较的元素的开始位置，以后称其为 “未处理部分”)，Y 指示从前 PMM 接受的元素的总数(根据情况以后称其为 “前向插入数”)，而 Z 是在归并前 PMM 和后 PMM 时所得到的虚拟数组中的前 PMM 给出的用于元素的建议顺序号(根据情况称为 “虚拟顺序号”)。

之后，由前 PMM 的处理器执行初始数据传送。在此数据传送之中，在 PUT 指针指示的位置处的元素经总线传送到后 PMM (见图 10B 和步骤 1303 和 1312)。注意在步骤 1302 中的分支中，在两个 PMM 之间进行处理时结果永远是 “是”，不过这一点将在下面讨论。在此

第一数据传送中，元素 “Carter”传送到后 PMM 。后 PMM 找出传送到的元素 “Carter”在存储于后 PMM 中的数组的部分的应该插入的位置(步骤 1313)。这并不需要数值实际插入，而是只要找到元素该插入的位置就足够了。在此具体实施方式中，存储于数组部分中的元素实际上是置于排序状态。因此，此插入位置可藉助于平分法或其他高速搜索法找出。找到插入位置就可以识别位于插入位置之前的顺序未定的元素范围(以后称其为 “范围 1”)。注意在此具体实施方式中，这里有一个约定，在元素是同一元素的场合，后 PMM 的顺序优先。所以，在由前 PMM 传送来的元素 “Carter”在后 PMM 中存在时，当存储于后 PMM 中时此顺序优先(即具有较小的顺序号)。

在此示例中，可以看到由前 PMM 传送来的元素 “Carter”位于由后 PMM 管理的数组部分内元素 “Carter”之前，于是可以看到属于范围 “1”的元素不存在(见图 10C 和步骤 1314 中的 “是”)。之后，后 PMM 的处理器将传送来的元素 “Carter”的顺序号设置为 “0(即开始处)”并将其返回给前 PMM(步骤 1315)。之后，后 PMM 的处理器使前向插入数递增到 “1”并且也将虚拟顺序号增加到 “1”(步骤 1316)。这是因为，由于从前 PMM 传送的元素数增加了 1，就必须使前向插入数递增，并且下一个元素的顺序号至少就此时给出的数值(在此场合为 “0”) 必须递增。

当顺序号(元素插入位置)由后 PMM 给定时(步骤 1332)，前 PMM 的处理器将给定的顺序号作为所述该元素的顺序号存储(步骤 1334)，并且之后使 PUT 指针递增(见图 11A 和步骤 1335)。这样，就确定了前 PMM 内的元素的次序。

之后，前 PMM 的处理器经总线将在 PUT 指针指示的位置的元素 “Smith”传送到后 PMM(见图 11B 和步骤 1303)。与前面传送元素 “Carter”的方式相同，后 PMM 找寻所传送的元素 “Smith”要插入的位置(步骤 1313)。可以看到，元素 “Smith”是位于由后 PMM 管理的数组部分中的元素 “Monroe”之后(见图 11C 和步骤 1314 中的 “是”)。从而，在由后 PMM 管理的数组部分中可以确定元素

“Monroe”和位于其前的元素的数，以及各个元素的次序。特别是，后 PMM 的处理器通过下面的进程确定上述元素的次序。

首先，将前向插入数 “Y”加到属于包含在范围 “1”内的元素的每个顺序号上(步骤 1317)。从而可确定包含在范围 “1”内的元素的次序。在上述示例中，元素 “Carter”的顺序号变为 “0+1=1”，而元素 “Monroe”的顺序号变为 “1+1=2”。之后，将包含在范围 “1”内的元素中的最后元素的顺序号替换虚拟顺序号(步骤 1318)，未处理位置变为范围 “1 ”中最后元素后面下一个元素的位置(步骤 1319)。在上述示例中，元素 “Monroe”的顺序号 “2”是在比较指针(结构数组)的 Z 中给出，并且未处理位置从 “0”变为 “2”。从而结构数组变为(2，1，2)。

在此进程之后，结构数组内的前向插入数 “Y”和虚拟顺序号 “Z”递增(步骤 1320)。从而结构数组变为(2，2，3)(见图 11D)。在步骤 1320 中取得的虚拟顺序号变为在步骤 1312 中取得的元素的顺序号(在上述示例中，为 “Smith”)，并且后 PMM 的处理器将所述该顺序号(在上述示例中，为 “3”)传送到后 PMM(步骤 1321)。在此进程之后，虚拟顺序号再递增(步骤 1322)。这是因为下一个元素的顺序号变为至少比此时给定的顺序号大 1 之故。

前 PMM 存储将接收到的顺序号作为所述该元素的顺序号存储并使 PUT 指针递增。这样就确定了前 PMM 中的顺序号。

当在前 PMM 中没有未处理的元素时(就是说，对所有的元素都确定了顺序号并且没有元素置于 PUT 指针的位置)，前 PMM 的处理器将一个指示结尾的值传送到后 PMM(见步骤 1306)。此处，“指示结尾的值”是一个大于指示数组的结尾处的元素的值。响应此上述指示结尾的值，后 PMM 执行一个几乎与此进程等同的进程(图 13B 中的步骤 1312-1322)。在上述示例中，不管是否接收到指示结尾的值，不存在包含在范围 “1”中的元素，所以处理经步骤 1315 和 1316 达到步骤 1323 而结束(见图 12B)。

在前 PMM 中，通过发送指示结尾的值(见步骤 1316)和确定所有

元素的顺序号(步骤1336中的 “是”)，处理结束。

PMM 14-3和PMM 14-4之间的归并的执行是藉助类似于上述进程的进程，并且于是每个元素的顺序号的确定就如图12C所示。

当在两个PMM中确定了每个元素的顺序号时，CPU 12改变开关的位置将每个都是由两个PMM组成的PMM组连接。图14A和图14B为示出属于示于图8中的存储器模块的两个存储器模块组之间的连接示例的示图。在图14A中，PMM 14-1和14-2组成第一PMM组，而PMM 14-3和14-4组成第二PMM组。CPU 12控制开关28和30使PMM 14-1和14-2的输出连接到PMM组14-3的输入，PMM 14-3的输出连接到PMM 14-4的输入，而PMM 14-4的输出连接到PMM 14-1和14-2的输出(见图7的步骤709)。另外一种方式是，如图14B所示，可控制开关使PMM 14-1和14-2的输出连接到PMM 14-3和14-4。

图15A和15B为分别示出示于图14A和14B的连接的示例的示意图。正如以后会了解到的，在图15A中，由PMM 14-4给到PMM 14-1和14-2的数据(图中的符号(1))指示顺序号，由PMM 14-1和14-2给到PMM 14-3的数据(图中的符号(2))指示元素，而由PMM 14-3给到PMM 14-4的数据(图中的符号(3))指元素和虚拟顺序号。此外，在图15B中，在PMM之间交换的数据(1)和(2)与图15A中的相同，并且另一方面从PMM 14-3传输到PMM 14-4的数据(符号(3))指示由PMM 14-3计算的虚拟顺序号。

下面描述归并两个PMM组的进程和根据两对PMM中的数组部分确定一个数组的顺序号和其中包含的元素的顺序号的进程，如上述图12A至12C所示(见图7的步骤709)。注意，在下面的描述中，每个PMM中执行的进程都是根据在图14B和15B中指示的连接状态描述的。

首先，分别在PMM 14-1和PMM 14-2中(以后称其为 “前PMM组”)，将PUT指针置于初始位置(步骤1301)。注意，在其后的进程中，组成前PMM组的PMM中每一个都根据它自己控制的元素的发送来移动PUT指针。另一方面，后PMM中的每一个在其结构数组初

始化时都将比较指针置于初始位置(步骤 1302)。之后，在组成前 PMM 组的 PMM 中的每一个之中，当前，组成前 PMM 组的 PMM 都确定被发送的元素的顺序号。

注意，在流程图中，发送时使用的发送指针和接收指针都用作 PUT 指针，但是根本上这些发送和接收指针的移动夹在后 PMM 组的处理时间之中，但是其执行只有微小的时间差。比如，如后所述，当在某个 PMM 中发送指针递增时(见步骤 1304)，所述该 PMM 也使接收进程中的接收指针递增(步骤 1335)。

组成前 PMM 的 PMM 确定所述该元素是否由其自己根据待处理的元素的顺序号进行控制(步骤 1302)。如步骤 1302 的结果是 “是”，就通过总线 24 将 PUT 指针指向的元素传送到 PMM 14-3 和 14-4(见图 13A 和图 16A 的步骤 1302)。在上述示例中，顺序号为 “0”的元素 “Carter”由 PMM 14-1 传送到 PMM 14-3 和 PMM 14-4。藉助这一进程 PUT 指针的位置在 PMM 14-1 中移动(步骤 1304)。

PMM 14-3 和 PMM 14-4 每个都接收元素(步骤 1312)，寻找这些元素应该插入的位置(步骤 1313)并确定属于范围 “1”的元素是否存在(步骤 1314)。对于上述的元素 “Carter”，步骤 1314 的结果是“否”。

因而，在 PMM 14-3 中，元素 “Carter”的虚拟顺序号变为 “0”，于是此值传送到 PMM 14-4。在 PMM 14-4 中元素 “Carter”的虚拟顺序号也变为 “0”。这样，PMM 14-4 的处理器经总线向前 PMM 组组返回 “MAX(0, 0)=0”作为元素 “Carter”的顺序号(见图 16B 和步骤 1315)。之后，在 PMM 14-3 和 14-4 中，结构数组内的前向插入数(Y)和虚拟顺序号(Z)每个都递增(步骤 1316)。在上述示例中，每个的结构数组就都变为(0, 1, 1), (0, 1, 1)。

当由后 PMM 组给出顺序号时(步骤 1331)，组成后 PMM 组的 PMM 确定当前正在处理的元素(比如，元素 “Carter”)是否是其自己控制的元素(步骤 1333)。在元素 “Carter”的顺序号传送时，PMM 14-1 发现上述步骤 1333 的结果是 “是”，并且利用由后 PMM 组给

出的顺序号代替与处于该位置的元素相应的顺序号(见图 16A 和步骤 1334)。

同样，后 PMM 组传送被赋予下一个顺序号的元素到后 PMM。在上述示例中，元素 “Carter”从 PMM 14-2 发送(见图 17A)，并且每个后 PMM 组的那些具有更高虚拟顺序号 “MAX(1，1)=1”的传送到前 PMM 组组作为所述该元素 “Carter”的虚拟顺序号(见图 17B)。此外，在组成后 PMM 组的 PMM 14-3 和 PMM 14-4 中，结构数组分别变为(0，2，2)和(0，2，2)(见图 17B)。

另外，前 PMM 组组传送被赋予下一个顺序号的元素到后 PMM。在上述示例中，元素 “Monroe”从 PMM 14-2 传送(见图 18A)。在 PMM 14-3 中，元素 “Monroe”确定为在由所述该 PMM 14-3 控制的元素 “Kennedy”的后面(见步骤 1313)。因此，在 PMM 14-3 中，由于元素 “Gore”和元素 “Kennedy”属于范围 “1”，前向插入数 “Y(=2)”分别加到元素 “Gore”和元素 “Kennedy”顺序号上。由此，元素 “Gore”的顺序号确定为 “0+2=2”，而元素 “Kennedy”的顺序号确定为 “2+2=4”(见步骤 1317)。之后，PMM 14-3 的处理器设定结构数组的虚拟顺序号 Z(当前值为(0，2，2))为范围 “1”内的结尾元素的虚拟顺序号 “4”(见步骤 1318)并使未处理位置前进(即将 X 的值从 “0”改变为 “2”)(见步骤 1319)。另外，PMM 14-3 的处理器使结构数组的前向插入数 “Y”和虚拟顺序号 “Z”递增(当前值是(2，2，4))(见步骤 1321)。从而此结构数组变成为(2，3，5)。PMM 14-3 中的虚拟顺序号 “Z(=5)”经总线传送到 PMM 14-4。其后，PMM 14-3 中的处理器使结构数组的虚拟顺序号 “Z”递增(见步骤 1322)。在上述示例中，结构数组通过执行步骤 1322 变成为(2，3，6)。

另一方面，在 PMM 14-4 中，元素 “Monroe”确定为位于所述该 PMM 14-4 控制的元素 “Johnson”和 “Wilson”中间(见步骤 1313)。因此，在 PMM 14-4 中，元素 “Johnson”属于范围 “1”，所以前向插入数 “Y(=2)”加到元素 “Johnson”的顺序号上，并

且这样元素“Johnson”的顺序号就确定为“1=2=3”(见步骤1317)。之后，PMM 14-4的处理器设定结构数组1虚拟顺序号“Z”(当前值为(0，2，2))为范围“1”内的结尾元素的顺序号“3”(见步骤1318)，并使未处理位置前进(即将X的值从“0”改变为“1”)(见步骤1319)。另外，PMM 14-4的处理器使结构数组的前向插入数“Y”和虚拟顺序号“Z”递增(当前值是(1，2，3))(见步骤1321)。从而此结构数组变成为(1，3，4)。

其后，PMM 14-4比较由PMM 14-3给出的虚拟顺序号“Z(-5)”与由它自己计算出的虚拟顺序号“Z(=4)”，求出更大值“MAX(5，4)=5”并将其传送到前PMM组组作为传送的元素“Monroe”的次序。从而在前PMM组组中(更具体地说，是在发送了元素“Monroe”的PMM 14-4中)，所述该元素的顺序号确定为“5”。注意，还是在PMM 14-4中，在步骤1321后，结构数组中的虚拟顺序号“Z”递增(见步骤1322)。在上述示例中，结构数组变为(1，3，5)。

元素“Smith”以同样的方式从前PMM组组发送(图19A)，但在此场合的进程也按照图13A-13C执行。为了再对其简略描述，接收元素“Smith”的PMM 14-3在元素“Smith”的插入位置的前面没有属于范围“1”的元素存在，所以PMM 14-3将其结构数组内的虚拟顺序号“Z(=6)”传送到PMM 14-4。PMM 14-4在元素“Smith”的插入位置的前面也没有属于范围“1”的元素存在，所以它比较其结构数组内的虚拟顺序号“Z(=6)”与如此传送到虚拟顺序号“Z(=6)”，求出更大值“MAX(6，5)=6”并将其传送到前PMM组组作为传送的元素“Smith”的次序(见图19B和步骤1315)。在前PMM组中，发送元素“Smith”的PMM 14-1以接收到的顺序号(=6)改写与元素“Smith”相应的顺序号。注意，在PMM 14-3中，通过步骤1316，其结构数组变为(2，4，7)，并且，另一方面，在PMM 14-4中，通过步骤1316，其结构数组变为(1，4，6)。

这样，当前PMM组中的所有元素发送结束时，组成前PMM组的PMM之一将一个指示结束的值传送到后PMM组(见步骤1306)。

组成后 PMM 组的 PMM 中的每一个都接收这个值并执行步骤 1312 和 1323 的处理。在上述示例中，次序未定的元素 “Wilson”存在于 PMM 14-4 中。由于这一原因，在 PMM 14-4 中，当步骤 1314 的结果为 “是” 时，前向插入数 “Y”加到属于范围 “1”的元素 “Wilson”的顺序号上以给出 “3+4=7”，并且这样得出的 “7”就设定为元素 “Wilson”的顺序号。在通过这一进程之后，组成后 PMM 组的 PMM 中的每一个都在步骤 1323 中得到结果 “是”，并且进程也在后 PMM 组中结束。

在上述示例中，数组内的元素是存储于 4 个 PMM 中，但是即使是在元素是存储于大量 PMM 中的情况下，准备各由 4 个 PMM 组成的附加的 PMM 组对并在这些组对之间执行类似的进程也就足够了。比如，考虑如图 20 上述的情况，其中某一个数组内元素是存储于 1024 个 PMM 中。在此场合，首先 PMM 1 和 PMM 2，PMM 3 和 PMM 4，PMM 5 和 PMM 6，...，PMM 1023 和 PMM 1024 互相连接(见 PMM 之间的实线)，并且元素的顺序号在这些两个 PMM 之间确定，之后由 PMM 1 和 PMM 2 组成的一对 PMM 组形成前 PMM 组，PMM 3 和 PMM 4 形成后 PMM 组，由 PMM 5 和 PMM 6 组成的一对 PMM 组形成前 PMM 组，PMM 7 和 PMM 8(未示出)形成后 PMM 组，...，由 PMM 1021 和 PMM 1022 组成的一对 PMM 组形成前 PMM 组，PMM 1023 和 PMM 1024 形成后 PMM 组，并且这些组对互相连接(见虚线)，而元素的顺序号在组成这些组对的 PMM 组之间确定。之后，由 4 个 PMM 组成的 PMM 对形成前 PMM 组，而其后的 4 个 PMM 形成后 PMM 组(见点划线)，并且之后由 8 个 PMM 组成的 PMM 对形成前 PMM 组，而其后的 8 个 PMM 形成后 PMM 组(见点线)，并且依此类推顺序形成各由 2^n 个 PMM 组所组成的 PMM 组的组对，之后在其间确定元素的顺序号。最终，在 1024 个 PMM 内的所有元素的顺序号可通过确定由包含 512 个 PMM 的前 PMM 组和包含 512 个 PMM 的后 PMM 组的 PMM 组对内的元素的顺序号而确定。

这样，通过形成各由 2^n 个 PMM 组所组成的 PMM 组的组对和其

后确定存储于组成组对的PMM组的每个PMM中的元素的顺序号(见图7的步骤709和710)，最终可确定所有元素的顺序号(在步骤710中的结果为 “是”)，并且其后，如需要，可根据上述顺序编号方式执行数组重整进程(见步骤711)。这一进程不是强制性的，不过通过生成其中元素按照顺序号排列的数组以后可以高速执行待执行的信息处理。

更具体地说，CPU 12控制开关28和30以使个PMM的输入和输出由某一总线连接。图21为示出在4个PMM之间的连接情况的示图。之后，PMM 14-1至PMM 14-4的处理器根据所取得的顺序号向总线释放元素和顺序号。每个处理器都监视释放到总线的元素及其顺序号，获取具有顺序号与原来由其固有 RAM 核心管理的元素下标(记录号)相同的顺序号的元素，并将其存储于 RAM 核心的合适区域。比如，对于原来将带有下标(记录号) “0”和 “1”的元素存储于其固有RAM核心(比如，见图10的PMM 14-1)的PMM，获取标记有顺序号“0”和 “1”的元素并将其存储就可以了。这样做，每个PMM就可以管理实际存储的数组。注意，这样一来，PMM 的处理器在生成存储数组时还将生成空间ID控制表。

另外一种方式是，如图22所示，令其他PMM(PMM 14-5至PMM 14-8)管理所存储的数组以使其他PMM中的每一个都监视顺序输出的元素及其顺序号，根据顺序号获取要由其自己存储的元素并将其存储于每个PMM的RAM核心中就足够了。

比如，考虑利用上述本发明提供1024个PMM，在每个PMM中存储大约1百万个数据(元素)并且执行此数据排序的场合。在此场合，排序完成需要的时间如下。此处，我们假定连接PMM的总线中的每一根都具有6.4 GB/s的数据容量，在处理时所有的PMM都并行运行(即不存在不执行进程的PMM)，并且所有相关的PMM都可协同并同时运行。此外，假设在每个PMM中完成大约1百万个数据(元素)的排序的时间为2.5秒。在此场合，可以看到对包含在1024个PMM中的大约10亿个元素进行排序只需要大约4秒钟。

藉助于这一具体实施方式，在开始时 PMM 分割为由 2 个 PMM 组成的对，之后在进一步分割为各由 2^n 个 PMM 组所组成的 PMM 组，并且之后确定在每个对之中的顺序号。另外，通过利用开关等等来调节应用于每个对的总线，就可以在各对之间并行执行顺序号的确定。此外，通过重复自前 PMM 组向后 PMM 组发送元素的进程和根据后 PMM 组内的结构数组中的值建立顺序号，可以在所有的对中建立顺序号。因此，处理可以以极端并行的方式执行而不会生成任何不执行任何处理的 PMM(所谓的 “空闲”PMM)，并且也可以减少利用总线传送的数据量。这就允许显著提高排序的速度。

注意，在上述具体实施方式 1 中，如图 14B 和 15B 所示，排序是通过连接 PMM 和在其中间向元素赋予顺序号而执行的，不过 PMM 也可以连接成如图 14A 和 15A 所示。在此场合，图 13B(步骤 1312 至 1323)中的后 PMM 组的处理不是并行执行的，而是每次在某一个 PMM 中取得一个虚拟顺序号，待处理的元素和所述该虚拟顺序号是传送到邻接的 PMM，并且步骤 1312 至 1323 是在此 PMM 中执行。因此，随着组成后 PMM 组的 PMM 的数目的增大，有时候这会导致处理产生相当的延迟。

[其他排序(具体实施方式 2)]

下面对本发明的具体实施方式 2 予以描述。在上述具体实施方式 1 中，所有元素(在前 PMM 组内的元素)都是传送到后 PMM 组。然而，随着数组的增大，可能出现大量的重复值。在采用根据具体实施方式 1 的技术时，取同样数值的元素被多次发送到总线。依情况的不同，可以认为反复发送具有同一数值的元素是一种浪费。于是，在具体实施方式 2 中，是预先计数 PMM 组内的元素数，并且通过在向后 PMM 组发送元素的同时一起发送元素数，可以防止经总线反复发送重复元素。

比如，考虑 4 对 PMM 的排序已经完成并且这些对相互连接来执行 8 个 PMM 的排序的场合。在此场合，如图 23 所示，最好是能够利

用一根用来执行 8 个 PMM 的归并和排序的总线(此总线位于图 23 中的 PMM 的下方；见符号 2301 至 2303)并且在元素之间交换数据。下面描述在如图 23 所示的连接模式中 PMM 14-1 至 PMM 14-4(以下，为方便起见，“PMM 14-1”至“PMM 14-4”分别称为“PMM 1”至“PMM 4”)中的数值的冗余度的计算。此处，连接 PMM 1 至 PMM 4 的输入/输出终端(I/O)的总线(见符号 2304)称为第 1 总线，而连接 PMM 1 至 PMM 4 的其他输入/输出终端(I/O)的总线(见符号 2305)称为第 2 总线。第 1 总线用来交换用于 PMM 1 至 PMM 4 组成的 PMM 组的信息，而第 2 总线用来向各 PMM 给出数值及其冗余度。

注意，在下面的描述中，如图 25 所示，在 PMM 1 至 PMM 4 的数组内计算对于它们顺序号已经附加到每个元素的那些(值)的冗余度。就是说，只对前 PMM 组计算冗余度就足够了。图 24 为示出在 PMM 中计算冗余度的进程的流程图。PMM 1 至 PMM 4 的每一个都首先执行各初始化进程(步骤 2401)。此处，每个 PMM 具有一个顺序号计数器用来指示待处理的值(元素)，一个等同值计数器用来指示某一个数值(元素)有多大的冗余度，以及一个用来保持待处理的前值(元素)的前值存储寄存器，并且将顺序号计数器和等同值计数器的初始值设定为“0”(见图 25).

其次，每个 PMM 都参照此顺序号计数器来确定待处理的元素的顺序号，并确定应用此顺序号的元素是否是它自己管理的元素(步骤 2403)。在上述示例中，顺序号计数器的数值的初始值为“0”，所以 PMM 3 确定它自己管理的一个元素有待处理(步骤 2403 中的“是”)。注意，其后的步骤 2404-2405 在初始处理(即对具有顺序号为“0”的元素的处理)中不考虑。PMM 3 确定在其自己内部有多少与具有顺序号为“0”(在此场合，“Carter”)的元素等同的元素(即 PMM 3 管理多少等于“Carter”的元素)，并将元素“Carter”发送到第 1 总线，以及本地 PMM 内部计数，此计数指示这些元素中它有多少(步骤 2406)。另一个 PMM (PMM 1，PMM 2 和 PMM 4) 在步骤 2403 中得到的结果是“否”，于是就前进到步骤 2407。

每个 PMM 都接收经过第 1 总线给出并根据本地 PMM 内部计数的数据，将 PMM 内部计数加到顺序号计数器的数值上(步骤 2408)。在上述示例中，顺序号计数器的数值变为 “0+1=1”。之后，确定给定值是否与前值存储寄存器中的数值不同(步骤 2409)，并且如两者等同，则本地 PMM 内部计数加到等同值计数器的数值上(步骤 2410)，另一方面，如是一个新的数值，就执行后述的替换进程(步骤 2411)。注意，在此初始迭代中，前值存储寄存器没有数值，所以上述的步骤 2409 省略，而元素存储于前值存储寄存器中，并且等同值计数器也递增。所以，在上述示例中，每个 PMM 在前值存储寄存器中存储所接收到的元素 “Carter”，并将等同值计数器设定为 “0+1=1”(见图 26)。

重复这些步骤 2401 至 2411 的处理并且当最后元素的处理完成时，步骤 2401 的结果为 “是”，于是处理前进到步骤 2412。

在上述示例中，当第 1 步骤 2401 至 2411 的处理完成时，每个 PMM 都参照顺序号计数器的值并确认此值为 “1”。从而 PMM 4 了解它管理带有顺序号为 “1”的元素。此外，PMM 4 比较前值存储寄存器的值(元素 “Carter”)和具有顺序号为 “1”的元素 “Carter”(步骤 2404)，并且值不改变，所以它通过第 1 总线发送元素 “Carter”和本地 PMM 内部计数 “1”(步骤 2405)。如图 27 所示，经过第 1 总线接收数据的 PMM 使顺序号递增(1+1=2)(步骤 2408)，并且因为存储于前值存储寄存器中的值与接收到的元素等同，它们使等同值计数器递增(1+1=2)(步骤 2410)。

之后，在每个 PMM 中执行对带有顺序号 “2”的元素的处理。在对带有顺序号 “2”的元素执行处理中，PMM 1 保持一个元素，于是 PMM 1 比较元素 “Clinton”与存储于前值存储寄存器的元素 “Carter”。此处值有改变(在步骤 2404 中结果为 “是”)，所以 PMM 1 通过第 2 总线发送前值存储寄存器的内容(元素 “Carter”)和等同值计数器的 “2”(步骤 2405)。寄存器的这一内容和计数器的值对各 PMM 给出。如后所述，当计算出某一元素(在此场合为元素

“ Carter ”)的冗余度时，执行所述该元素的排序(?)(见图 31A 至 31C)。因此，此元素及其冗余度可保持于个 PMM 中，一直到所述该元素的排序完成。

此外，元素 “ Clinton ”及本地 PMM 内部计数 “ 1 ”给到第 1 总线(步骤 2406)。

根据经由第 1 总线给出的数据，PMM 使顺序号计数器递增(2+1=3)(步骤 2408)。此外，前值存储寄存器的值 “ Carter ”不同于传送的元素 “ Clinton ”(在步骤 2409 中结果为 “ 是 ”)，所以 PMM 替换(更新)前值存储寄存器的值并且也替换等同值计数器的值为经第 1 总线给出的本地 PMM 内部计数的值(见步骤 2411 和图 28A)。

对带有其他顺序号的元素执行同样的进程。比如，对顺序号 “ 3 ” PMM 3 根据步骤 2404 和 2406 经过第 1 总线发送元素 “ Clinton ”并且根据步骤 2407，2408，2409 和 2410 使各计数器递增(见图 28B)。此外，对顺序号 “ 4 ”PMM 1 根据步骤 2404，2405 和 2406 经过第 2 总线发送元素 “ Clinton ”并且根据步骤 2407，2408，2409 和 2411 使各计数器递增(见图 29A)。

在处理顺序号 “ 5 ”时，存在两个由 PMM 2 管理的 “ Johnson ”，所以它经过第 1 总线发送元素 “ Johnson ”和本地 PMM 内部计数 “ 2 ”。因此， “ 2 ”加到顺序号计数器和等同值计数器的值上(见图 29B)。此外，藉助此进程，顺序号计数器的值从 “ 5 ”改变为 “ 7 ”，因此，注意待处理的下一个元素的顺序号变为不是 “ 6 ”，而是 “ 7 ”。当具有顺序号 “ 7 ”(见图 30A)的最后一个元素的处理完成时，步骤 2401 的结果为 “ 是 ”。于是，开始的 PMM(在上述示例中为 PMM 1)，通过第 2 总线，发送元素 “ Johnson ”和等同值计数器的数值 “ 4 ”并经过第 2 总线发送指示处理已经结束的数据(步骤 2414)。每个 PMM 都经过第 2 总线给予每个元素和指示元素数的计数，并且将这些应用于排序。注意，在上述示例中，开始的 PMM 的构成使得他执行步骤 2413 和 2414，不过这并不是限制，因为对 PMM 而言预先规定最后元素等等，以及指示结束的数据就足够了。

如上所述，通过取得在某一 PMM 组内的现有各元素的数目，在将一个 PMM 组和另外一个 PMM 组归并并对其排序时，不需要发送重复的元素。

图 31A 至 31C 为示出在排序同时省略发送冗余元素的的流程图。图 31A 至 31C 与图 13A 至 13C 等同，只有某些例外，所以后两位数字相同的步骤是大致相当的进程。此外，在图 31A 至 31C 中有双线环绕的步骤是新添加的步骤或是与图 13A 至 13C 中的相应步骤略有不同的步骤。在此进程中，在前 PMM 组中，管理待处理元素(即由发送指针指向的元素)的 PMM 向后 PMM 组发送该元素同时一起发送前 PMM 组中的该元素的冗余度(数存在) “N”(见步骤 3103 和 3103-2)。在示于图 25 至 30B 的示例中，例如，当元素 “Carter”从 PMM 1 至 PMM 4 组成的前 PMM 组发送到后 PMM 组时，元素 “Carter”是与前 PMM 组中的元素 “Carter”一起传送的。此外，在前 PMM 组的发送进程中，输出元素及其冗余度的 PMM 在输出后将输出指针移动，移动数等于它自己管理的所述元素的数(步骤 3104)。比如，如图 28 所示，元素 “Carter”的冗余度是 “2”，并且这些在 PMM 3 和 PMM 4 中各管理一个。因此，第 2 指针的位置在 PMM 3 和 PMM 4 中各下降一个。注意，在 PMM 中的发送指针的移动和等于所述该元素的冗余度 “N”。

另一方面，组成接收元素及其冗余度的后 PMM 组的 PPM 将冗余度 “N”添加到前向插入数和虚拟顺序号，如图 31B 中的步骤 3116 和步骤 3120 所示。这相应于有 “N”个元素(带有小顺序号)位于其自身前面的场合。

此外，在组成后 PMM 组的 PMM 的接收进程中，在前 PMM 的发送进程中发送的元素(待比较数据)和在处理中由后 PMM 发送的顺序号用于计算接收到的顺序号和待比较数据发送时的顺序号的差值 “M”(步骤 3132-2)。这一差值 “M”指示位于待比较元素之前的元素数(如顺序号小于所述该元素)。因此，组成后 PMM 组的 PMM 在由其自己管理的元素中识别与所述该待比较元素等同的元素(步骤

3132-3)，并且如存在，就在只写元素各自的顺序号上加上 “M”(步骤 3134)。在步骤 3134 之后，PMM 移动接收指针，移动数等于所述元素数(步骤 3135)。这一进程大致与步骤 3104 等同。

下面描述示于图 24 的冗余度计算中和示于图 31A 至 31C 的排序(依情况有时称为自排序)中的并行操作。如图 23 所示，在此具体实施方式中，在冗余度计数中涉及的 PMM 之间的通信是利用总线 2304 和 2305 执行的，而在执行自排序中涉及的 PMM 之间的通信是利用总线 2301，2302，2303 等等执行的。为此目的，如在 PMM 中并行处理可行，则冗余度计数和自排序可以并行执行。在此场合，当在前 PMM 组中的某一个元素的冗余度计算(比如，如图 28A 所示)，元素 “Carter”及其冗余度 “2”经由第 2 总线发送并且当由组成前 PMM 组的 PMM (PMM 1 至 PMM 4)接收到时，在图 31A 至 31C 中所示的进程可对计算其冗余度的元素执行。就是说，响应对某一元素的冗余度的计算，可以执行对所述该元素执行步骤 3102-3104，步骤 3112-3122 和步骤 3132-3135 的处理。此外，在示于上述图 31A-31C 的处理中间，某一元素及其冗余度可以一起删除而完成针对所述该元素的那些进程。因此，在组成前 PMM 组的每个 PMM 中，并非所有与元素及其冗余度(其量随着不同元素数的增加而增大)有关系的数据都需要保留。

这样，在具体实施方式 2 中，后 PMM 组计算冗余度并发送元素及其冗余度到后 PMM 组。因此，前 PMM 组必须冗余地发送同一元素到后 PMM 组。在冗余等同元素数大的场合(比如，当一个元素指示人的性别，或人的年龄等等)，可以减少自排序处理的次数，于是排序可以以更高的速度进行。

[汇编(具体实施方式 3)]

下面描述本发明的具体实施方式 3。在具体实施方式 3 中，根据由置于各 PMM 内的元素组成的数组，形成(？)记录，无重复元素的数值表，以及用来由记录指定数值表的指针数组。在此说明书中，这一进

程称为汇编。

比如，当某一数组的元素在4个PMM(PMM 1至PMM 4)之间分割时，将PMM按照图32所示连接就足够了。如图32所示，PMM 1至PMM 4的输入/输出引脚(I/O)由第1总线连接(见符号3201)，而另一方面，PMM 1至PMM 4的输出引脚(O)和另一PMM “k”的输入引脚(I)连接到第2总线(见符号3202)。

第1总线用于与PMM 1至PMM 4组成的PMM组交换信息，而第2总线用于将元素及其冗余度给予另一PMM “k”。在此具体实施方式中，根据上述元素及其冗余度，在另一PMM “k”中形成数值表和计数数组等等。注意，此PMM “k”可以是PMM 1至PMM 4以外的PMM，当然也可以是PMM 1至PMM 4中的一个。

图33为示出根据本具体实施方式3的汇编的流程图。注意，为了简化描述，考虑元素在PMM 1至PMM 4之间如图34A那样分割并且其间顺序号的赋予进程已经执行。首先，每个PPM都有一个指示待处理的值(元素)的顺序号的顺序号计数器，一个指示处理后的所述该值(元素)的顺序号的值-号计数器，一个指示有多少所述该元素的冗余复件存在的等同值计数器以及一个保持待处理的值(元素)的前值的前值存储寄存器(见步骤3301和图34A)。注意，初始时前值存储寄存器无值保持。

之后，图33的步骤3302-3306的处理几乎与图24的步骤2401-2406等同。就是说，每个PMM都参照顺序号计数器，确定待处理的元素的顺序号，并确定应用此顺序号的元素是否是它自己管理的元素(步骤3303)。在图34A的状态，顺序号计数器的值为“0”，所以PMM 3生成一个本地PMM内部计数用来指示所述该PMM 3有多少在第2总线上的具有顺序号为“0”的元素(在此示例中为“1”)(见图34B的3306)。之后，PMM 3比较存储于前值存储寄存器中的元素与释放到第1总线的元素，并且如两者不同，就设定值-号计数器的值为好象它是发送到第1总线的顺序号(步骤3307)。注意，在图34A的状态中，值-号计数器的值是初始值“0”，所以应用于元素“Carter”的

顺序号不改变(见图 34B)。

之后，每个 PMM 3 都接收经过第 1 总线给出(步骤 3308)的数据。步骤 3308-3311 的处理几乎与图 24 的步骤 2408-2411 的处理等同。就是说，每个 PMM 将 PMM 内部计数加到给定的顺序号计数器的数值上，并且，如元素不是给定的数据中的新元素(步骤 3310 中为 “否”)，则将此 PMM 内部计数添加到等同值计数器的值上(见步骤 3311 和图 34B)。如图 34A 和 34B 所示，当带有顺序号 “0”的元素 “Carter”的处理完成时，带有顺序号 “1”的元素的处理以同样的方式执行(见图 35A)。

下面，执行带有顺序号 “2”的元素的处理。此处，PMM 1 比较存储于前值存储寄存器中的元素 “Carter”与带有顺序号 “2”的元素 “Clinton”。此处，它们是不同的(在步骤 3304 中结果是 “是”)，于是 PMM 1 经第 2 总线发送存储于前值存储寄存器之中的元素和等同值计数器的值(步骤 3305)。之后，PMM 1 经第 1 总线发送待处理的元素 “Clinton”和本地 PMM 内部计数 “1”(步骤 3306)。其后，PMM 1 比较存储于前值存储寄存器之中的元素和释放到第 1 总线的元素。在释放元素 “Clinton”的场合，这些是不同的，于是将元素 “Clinton”的顺序号设定为加上 “1”(0+1=1)的值-号计数器的值。

每个 PMM 接收经第 1 总线给出的数据(步骤 3308)并在接收到的数据中将本地 PMM 内部计数加到顺序号计数器的值之上(2+1=3)(见步骤 3309 和图 35B)。在给出元素 “Clinton”的场合，前值存储寄存器的元素 “Carter”与给出的元素 “Clinton”是不同的(步骤 3310 的结果为 “是”)，于是每个 PMM 执行新-值登记进程(步骤 3312)。在此进程中，值-号计数器的值递增(0+1=1)，等同值计数器的值改变为接收数据内的本地 PMM 内部计数 “1”，并且前值存储寄存器的内容重写为元素 “Clinton”(见图 35B)。

对带有顺序号 “3”的元素 “Clinton”执行同一进程。比如，PMM 3 可经第 1 总线发送元素 “Clinton”和本地 PMM 内部计数

“1”(见步骤3306)并且设定所述该元素 “Clinton”的顺序号为值-号计数器的值 “1”(见步骤3307和图36A)。此外，每个PMM在顺序号计数器的值上加上接收到的本地PMM内部计数 “1”(见步骤3309)并且也在等同值计数器的值上加上本地PMM内部计数 “1”(见步骤3311和图36A)。另外，如图36B所示，关于带有顺序号 “4”的元素 “Johnson”也一样，PMM 1经第2总线发送元素 “Clinton”和等同值计数器的值 “2”(步骤3305)，并经第1总线发送元素 “Johnson”和本地PMM内部计数 “1”(步骤3306)，并且设定元素 “Johnson”的顺序号为设定为加上 “1”(1+1=2)的值-号计数器的值。另一方面，每个PMM执行次序-值计数器递增和新-值登记进程(步骤3309和3312和图36B)。对带有其他顺序号的元素执行同一进程。各元素的处理示于图37A和图38A。注意，关于图38，PMM 1经第1总线发送最后一个元素 “Johnson”和那些元素的计数并经第2总线发送指示结束的数据(见步骤3315)。

如上所述，PMM “k”的输入与第2总线连接。因此，非冗余元素和与其相关的值-号计数器的值经第2总线给出。因此，PMM “k”接收它们并将接收到的严肃顺序地置于数值表中并且还将接收到的值-号计数器的值顺序地置于计数数组中。图39A为数值表和在PMM “k”内部生成的计数数组的示图。这些是在步骤3305或步骤3314中发送(见图35B，图36B和图38)并传送到PMM “k”。如图39A所示，元素置于无冗余数值表中，而指示每个现有元素数的计数(即冗余度)置于计数数组中。

另外，在PMM 1至PMM 4中可以生成一个指示记录和应用于每个无冗余元素的顺序号的指向数值表的指针数组。就是说，通过生成一个指示记录和应用于与所述该记录相对应的元素的顺序号之间的相应关系数组，可以使其成为指向此数值表的指针数组(见图39B)。在图39B中，关于记录 “0”，相应元素的顺序号 “2”变为指向数值表的指针数组中的指针值。这指示，在数值表中(见图39A)，待指向的数值是存储位置号是 “2”。就是说，根据指向数值表的指针数组中

的指针值，可指向存储于PMM “k”中的数值表，并且从而可以从记录识别元素。

藉助于如上描述的本具体实施方式，在PP之间分割的任何数组的元素被存储并被给予顺序号，并且此次序被重赋值以使同一次序赋予同一元素。求出元素和一个新取得的非冗余次序之间的对应关系并将元素存储于数值表中。所述该次序是指向数值表的指针数组和分割数组内的元素之间的对应关系。因此，可以根据记录通过指针数组内的指针值识别数值表内的元素。

[联结数值表(具体实施方式4)]

下面描述本发明的具体实施方式4。在具体实施方式4中，将两个数组联结。其前提是数值表和指向数值表的指针数组是通过汇编生成的。此外，一个空间ID应用于数值表和指向数值表的指针数组，并且有关由其自己管理的数组部分是由所述该空间ID管理。

图40为示出根据具体实施方式4的联结的流程图。

为简化描述，如图41A所示，考虑原始数据是一个由与在某个PMM组上分割的记录(见符号4100)相对应的元素所组成的数组的场合。汇编此记录组的结果是生成一个由指针数组(见符号4100)组成的数据块(以下称其为“信息块”)和一个在由PMM 1和PMM2组成的PMM组中的数值表(见符号4102)。另一方面，由对应于记录(见符号4110)的元素所组成的其他数组在其他PMM组之间分割，并且通过汇编，由指针数组(见符号4111)组成的信息块和数值表(见符号4112)在PMM 3和PMM 4组成的数组中生成。

CPU 12向每个PMM传送一个命令将数值表与指示两个数值表的数组的空间ID相联结的指令。在PMM之间，其中待联结的数组是它们自己管理的数值表，或其一部分，的PMM根据空间ID识别待联结的数值表(见步骤4001和图42A)。其次，PMM 1至PMM 4在两个数值表被归并并且将顺序号赋予元素的状态下对其进行排序(步骤

4002)。根据具体实施方式的排序可用于这一排序。在上述示例中，首先，元素的顺序编号是对由 PMM 1 和 PMM 2 组成的第一 PMM 组和由 PMM 3 和 PMM 4 组成的第二 PMM 组执行，通过取第一 PMM 组为前 PMM 组和取第二 PMM 组为后 PMM 组而将顺序号应用于两个组内元素。图 42B 为示出顺序号以这种方式应用于元素的状态的示图。

之后，对其中待处理的数值表是分割的 PMM 执行汇编以便在另外的 PMM 中或在 PMM 1 至 PMM 4 之一中生成一个联结数值表和联结计数数组(步骤 4003)。就是说，执行汇编可得到一个其中的归并数值表的元素不是冗余元素的新的数值表和一个包含指示存在有多少重复元素的计数的计数数组(见图 42C)。在这一进程之后，需要一个新的指针来指向联结的新数值表(即通过汇编得到的数值表)。这一点可通过生成一个存储与通过汇编得到的顺序号数组相对应的顺序号的新的指针数组而做到，上述顺序号指示在联结之前在与所述该指针数组内的指针值的位置相对应的位置的信息块中的指针数组内的指针值。在上述顺序号数组内的数值理解为与应用于元素的新顺序号相对应(见图 42C)。

如图 43A 所示，比如，在指示第一指针值 “1”的位置上的顺序号数组内的值(顺序号)是 “2”，于是在联结后在指针数组内的相应位置的指针值变为 “2”。此外，在指示第二指针值 “2”的位置上的顺序号数组内的值(顺序号)是 “3”，于是在联结后在指针数组内的相应位置的指针值变为 “3”。这样，就可以得到用于联结数值表的指针数组(图 43A 和图 43B)。

这样，可以了解，可利用新指针数组和联结数值表从记录识别此值(元素)。如图 44A 和 44B 所示，在新得到的指向数值表的指针数组内的相应的位置的指针值得到识别并且此指针值识别在由该值指示的位置上的数值表内的元素。此处，可以看到，同一元素的指定与在原始数据的元素中一样，即使是两个数值表已经联结。

这样，根据具体实施方式 4，通过归并多个数值表并且将对归并数值表的元素排序和汇编相结合，可以对每一个数值表得到一个联结

数值表和顺序号数组。顺序号数组的值(顺序号)可藉助用来从记录指定数值表的指针数组来识别并将所述该顺序号存储于与记录相应的位置，于是可根据记录得到新指针数组用来指定联结数值表。因此，就可以在大致为上述排序和汇编所需要的时间内将多个数值表联结，于是可显著增加处理速度。

本发明绝不限于上述具体实施方式，因为在权利要求的范围内可以有各种改型，并且，勿庸赘言，这些也包含于本发明的范围之内。

比如，虽然在上述具体实施方式中本发明是应用于计算机系统，但这并不是限制，因为本发明也可应用于连接到个人计算机等等的计算机电路板。在此场合，在图 1 中，CPU 12，存储器单元 14，总线 24 等等可安装于电路板上，从而组成根据本发明的信息处理单元。

此外，连接 CPU 12 和存储器模块 14，和/或总线之间的总线的组数不限于上述具体实施方式中的数目，而是可以考虑安装计算机系统的电路板的大小、各总线的位宽度等等适当地确定。此外，在上述具体实施方式中设置有用来设置存储器模块输入/输出和总线之间的连接的开关 28 及可以在 CPU 和存储器模块中间和存储器模块之间开关总线的开关 30。通过设置开关 29 和开关 30，比如，某一根总线(见图 1 的总线 24-4)既可用于 CPU 模块 12 和存储器模块 14-1 中间的数据交换，也可用于存储器模块 14-2 和存储器模块 14-3 之间的数据交换(在此场合将开关 30-5 关断即可)。因此，总线可有效地利用。然而，在总线组数可以足够大的场合或在存储器模块数相对小的场合，开关 30 不一定需要设置。

此外，本说明书叙述的是来自 CPU 12 的指令是通过信号控制线 25 给出，但是除了指令之外，当然时钟信号或各种用来使各存储器模块同步工作的其他控制信号也可以通过信号控制线 25 给出，并且也可以给出从各种存储器模块到 CPU 12 的规定信号(比如，出错信号和指示数据收到的信号)。

另外，在上述具体实施方式中，示出了 PMM 之间的连接的各种示例，但是选择用来在 PMM 之间进行连接的总线和信号交换并不限

于在上述具体实施方式示出的方式。

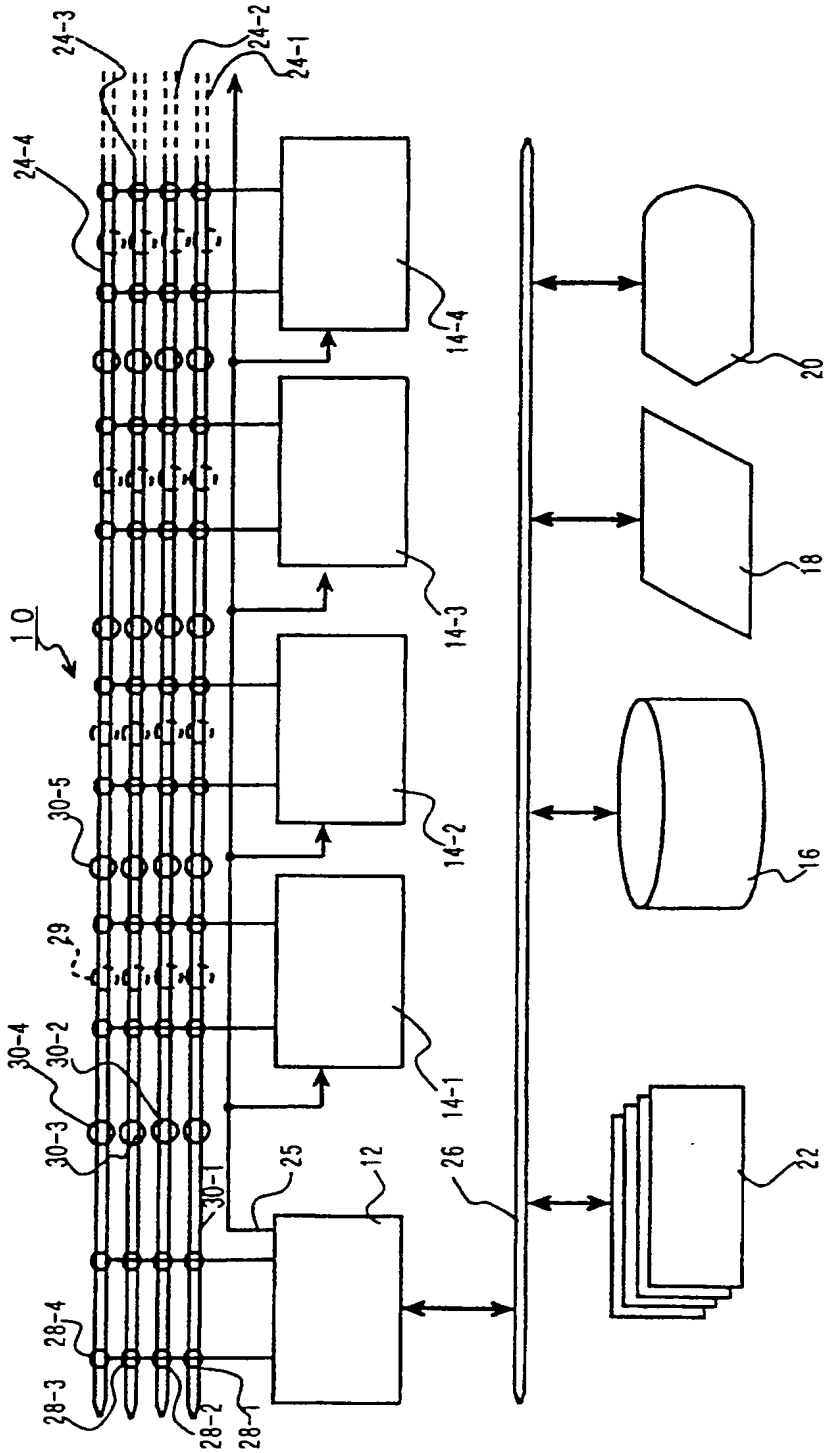
此外，在上述具体实施方式3中，如图32所示，第1总线(见符号3201)是用来实现PMM之间的通信，而第2总线(见符号3202)是用来实现元素和这些元素的计数(冗余度)，但是这并非是一种限制。比如，如图45所示，生成一个没有冗余元素的数组数值表的PMM“k”及其计数数组可能可以监视第1总线4501并根据出现在第1总线4501上的元素及计数数组实行规定处理(比如，由PMM 1至PMM 4执行的计数器递增和寄存器的内容的存储/更新)，从而生成数值表及计数数组。

另外，在此说明书中，一种装置的功能可能由两种以上的物理装置实现，或者是两种以上的功能可以由一种物理装置实现。

利用本发明，可以提供一种可以以极高速度执行数据排序、汇编和联结的信息处理装置。

本发明可应用于处理大量数据的系统，比如，数据库和数据仓库中。更具体说，本发明可应用于大规模科学技术计算和排序管理及对任务很关键的行政工作。比如证券交易。

图1


10
12
14-1
14-2
14-3
14-4
16
18
20
22
24-1
24-2
24-3
24-4
25
26
28-1
28-2
28-3
28-4
29
30-1
30-2
30-3
30-4
30-5

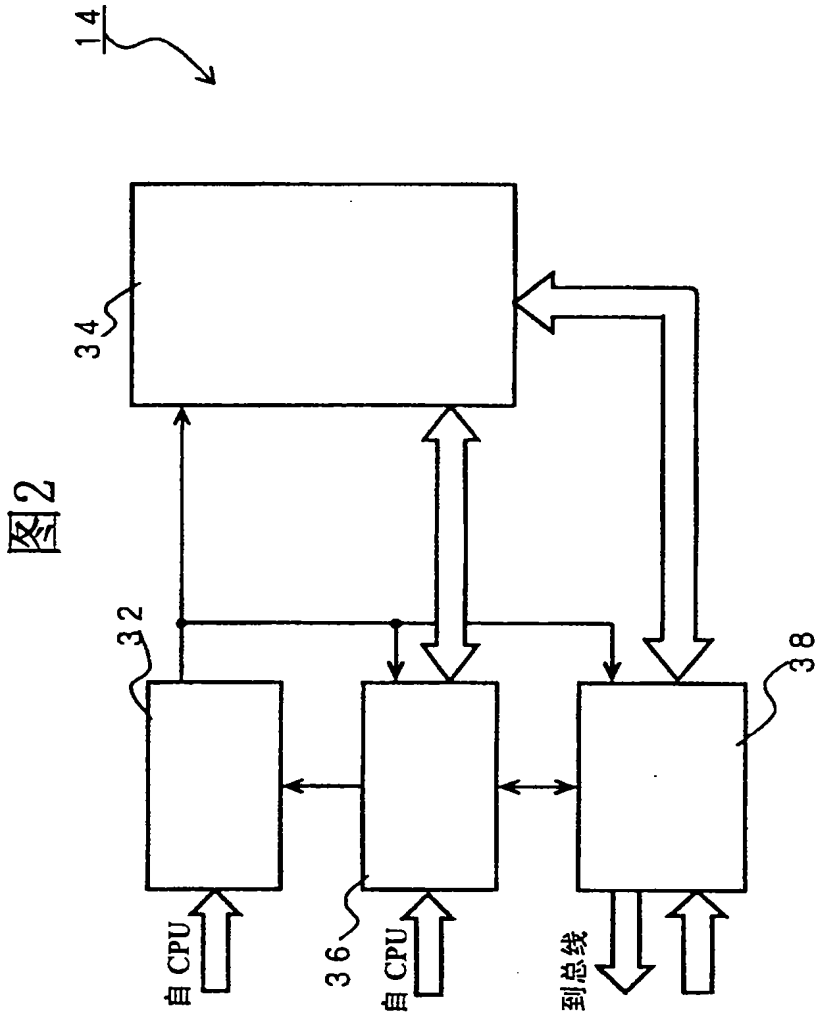


图2

图 3

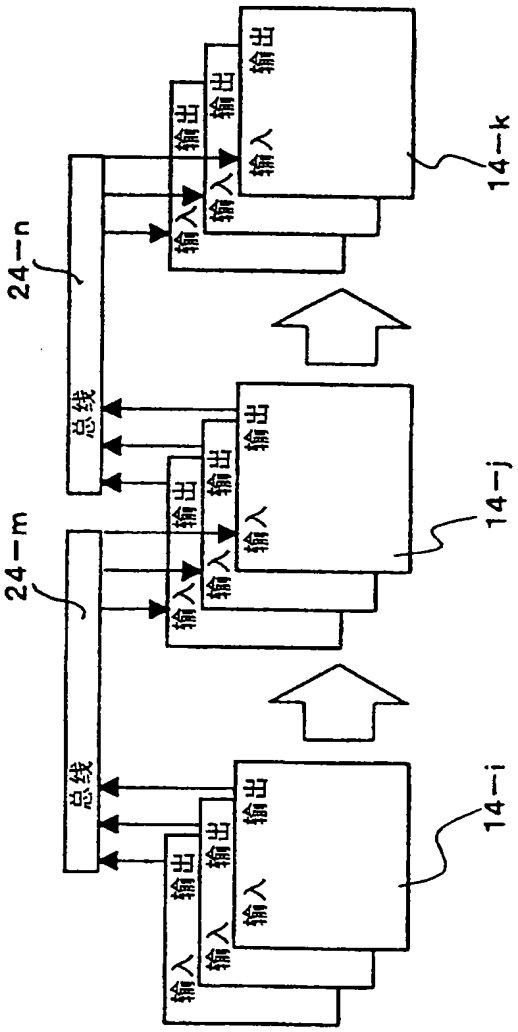


图 4A

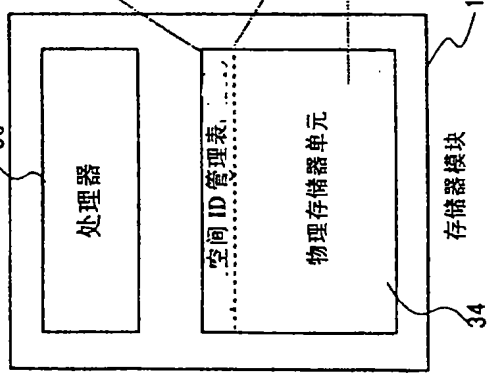


图 4B

空间 ID	逻辑起始地址	分配区域大小	片上物理起始地址	总量	访问控制标记
010	100	60	0	164	R
010	160	4	80	164	R
016	0	20	60	20	RW
…	…	…	…	…	…
…	…	…	…	…	…
212	0	100	360	100	RW
–	–	–	–	–	–
–	–	–	–	–	–

访问控制标记
R：只读
W：只写
RW：读写

图 4C

	物理地址	物理存储单元的内部 ID 逻辑地址	
0	10	010:0100	空间 ID 010
1	21	010:0101	
…	…	…	
59	33	010:0159	
60	2323	016:0000	空间 ID 016
…	…	…	
79	3241	016:0019	
80	15	010:0159	空间 ID 010
…	…	…	
83	8	010:0163	
…	…	…	
…	…	…	
360	-589	212:0000	空间 ID 212
…	…	…	
459	-1022	212:0099	
…	…	…	

图 5A

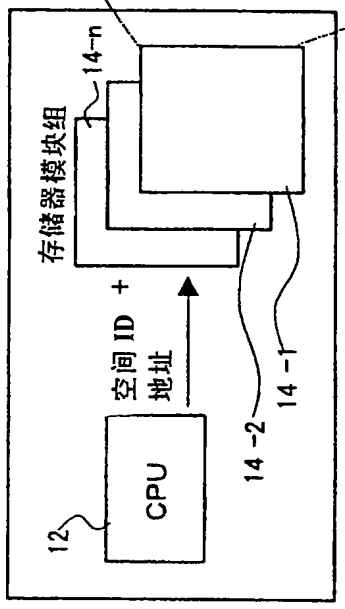


图 5B

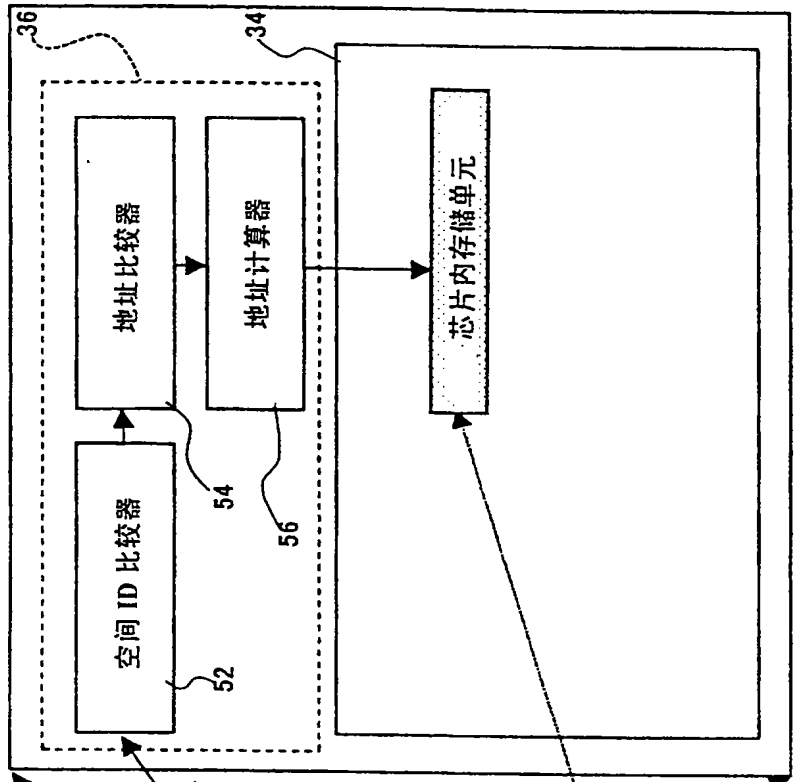


FIG. 5C

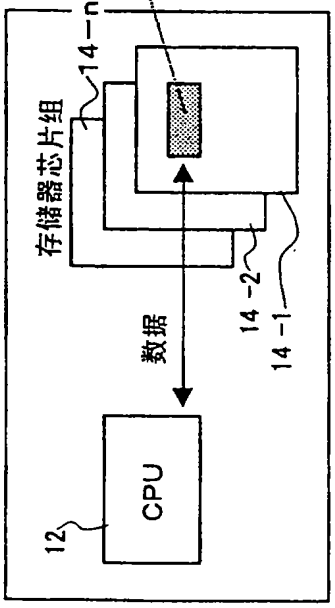
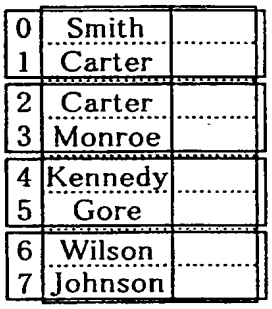


图 6A



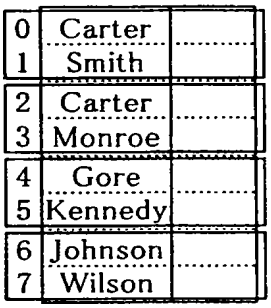
由 PMM 14－1 管理的范围

由 PMM 14－2 管理的范围

由 PMM 14－3 管理的范围

由 PMM 14－4 管理的范围

图 6B



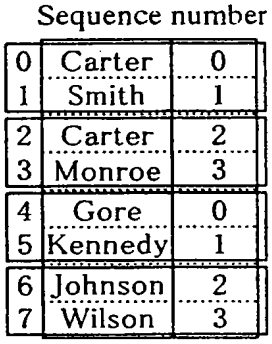
由 PMM 14－1 管理的范围

由 PMM 14－2 管理的范围

由 PMM 14－3 管理的范围

由 PMM 14－4 管理的范围

图 6C



由 PMM 14－1 管理的范围

由 PMM 14－2 管理的范围

由 PMM 14－3 管理的范围

由 PMM 14－4 管理的范围

图 7

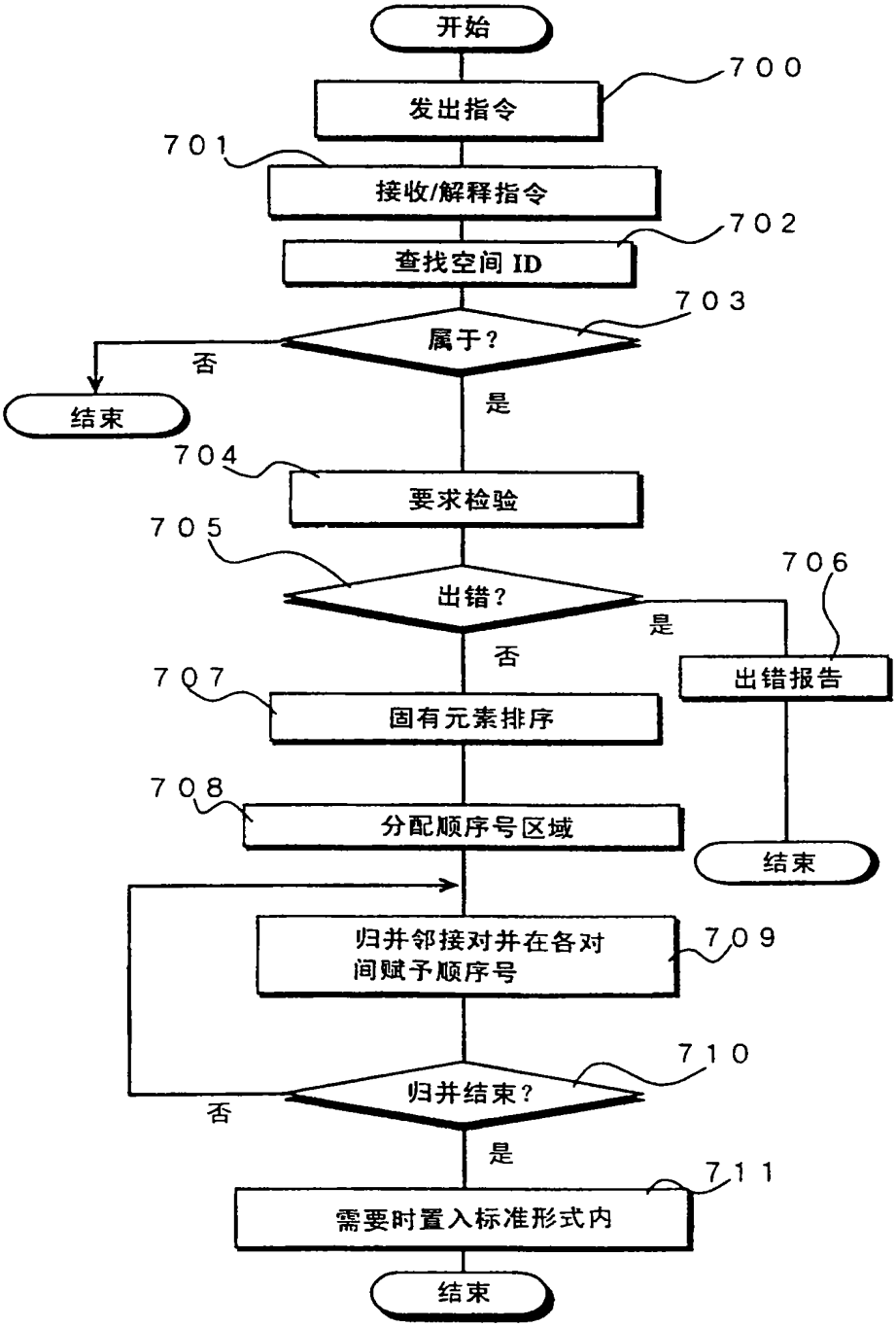

开始
700
发出指令
701
接收/解释指令
702
查找空间 ID
703
属于?
否
结束
是
704
要求检验
705
出错?
706
是
出错报告
否
707
固有元素排序
708
分配顺序号区域
结束
归并邻接对并在各对
间赋予顺序号
709
710
归并结束?
否
是
711
需要时置入标准形式内
结束

图 8

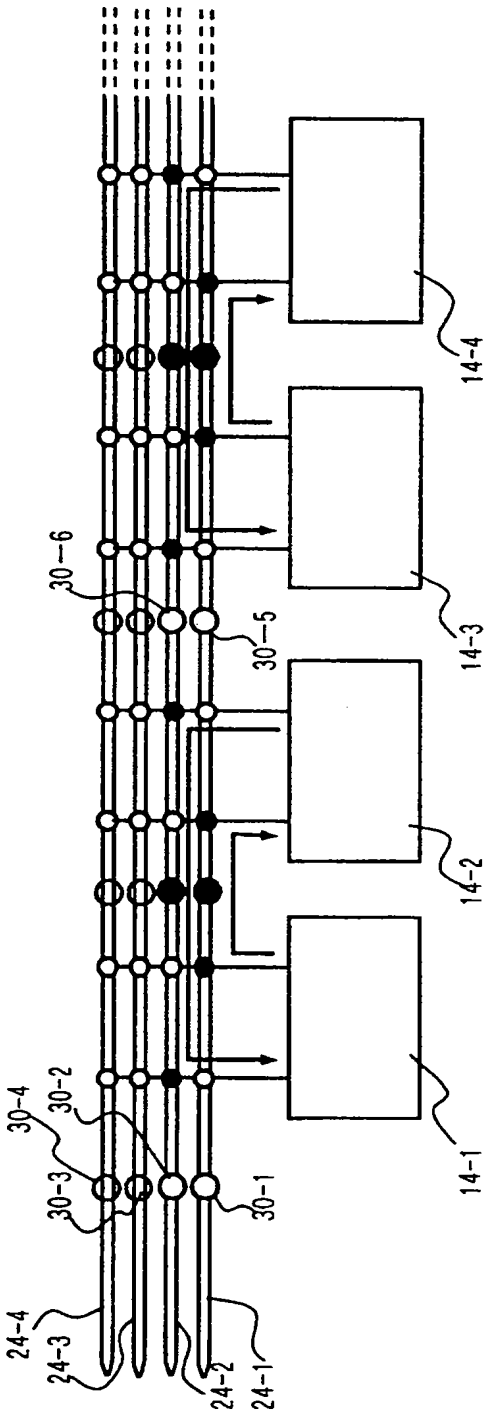


图 9

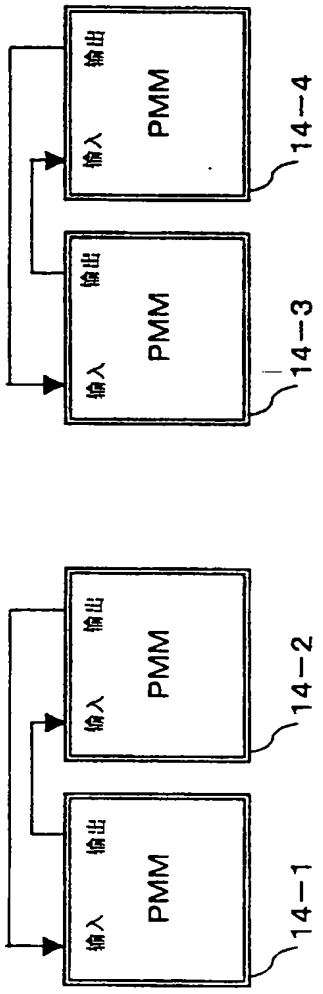


图 10A

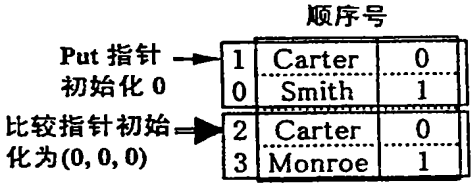


图 10B

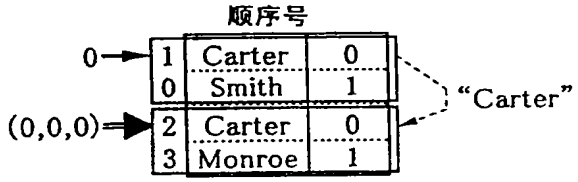


图 10C

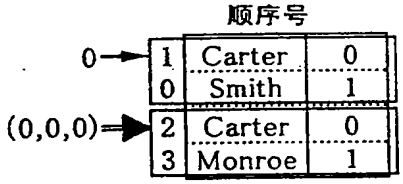


图 10D

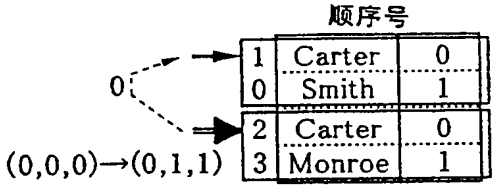


图 11A

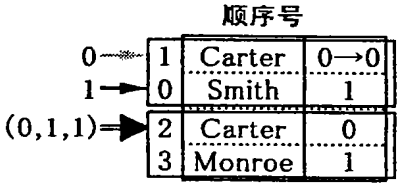


图 11B

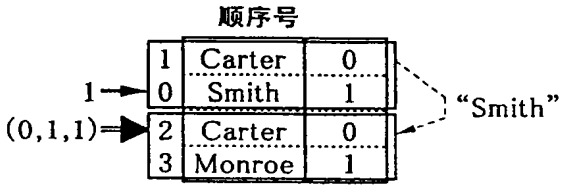


图 11C

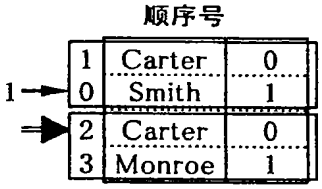
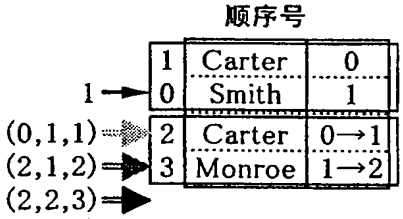


图 11D



(2,2,4) [步骤 1322 的处理之后]

图 12A

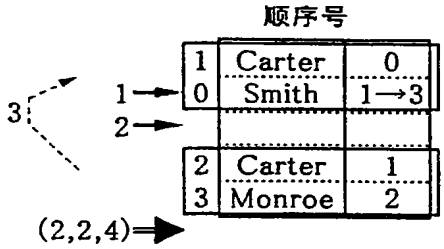


图 12B

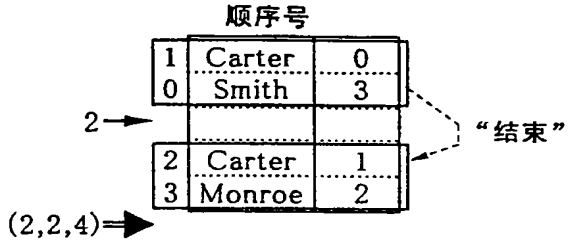


图 12C

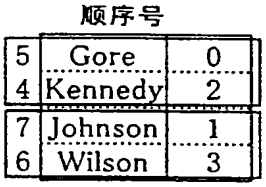


图 13A

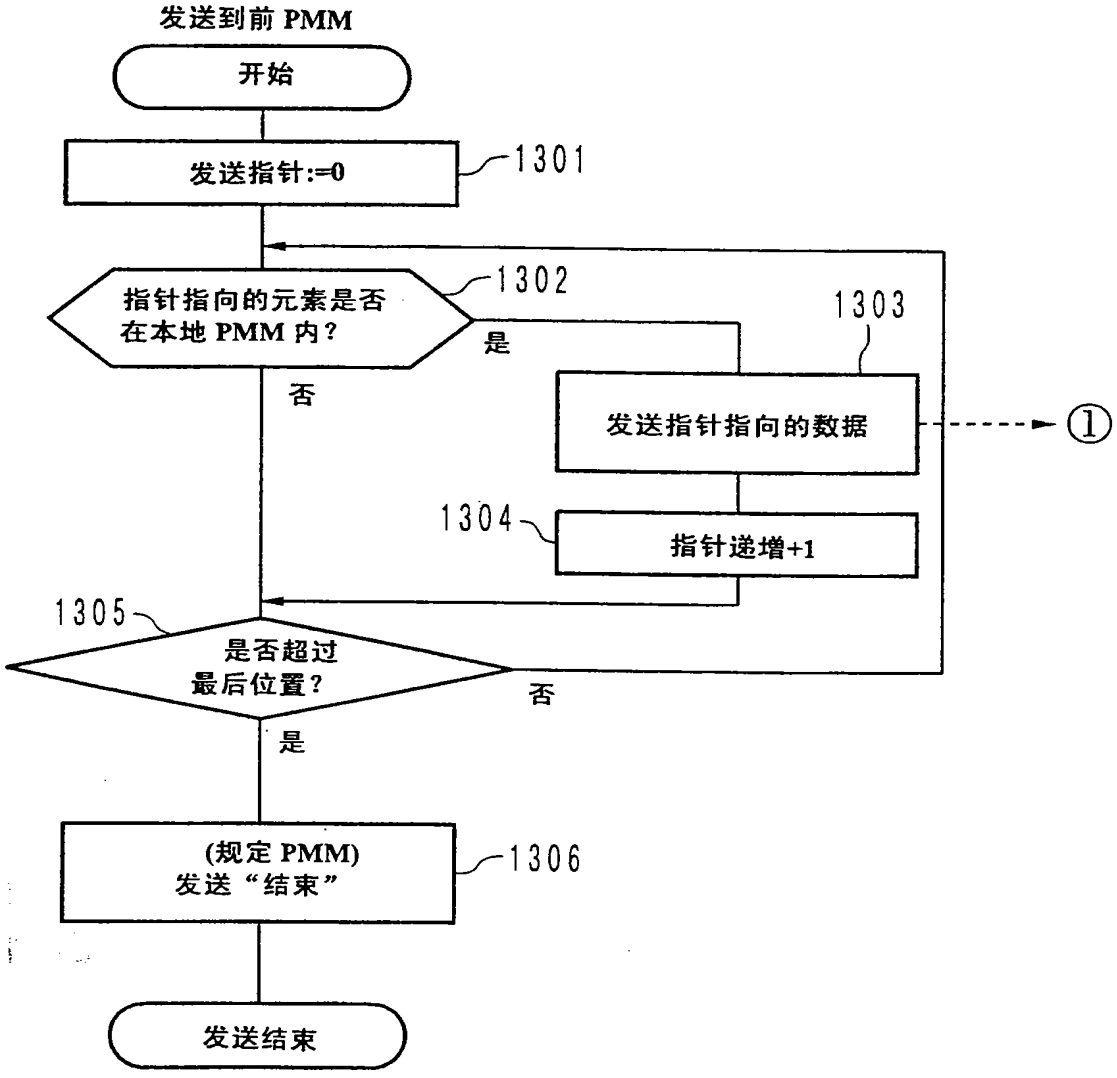

发送到前 PMM
开始
发送指针:=0
1301
指针指向的元素是否
在本地 PMM 内？
1302
是
否
1303
发送指针指向的数据
①
1304
指针递增+1
1305
是否超过
最后位置？
否
是
(规定 PMM)
发送“结束”
1306
发送结束

图 13B

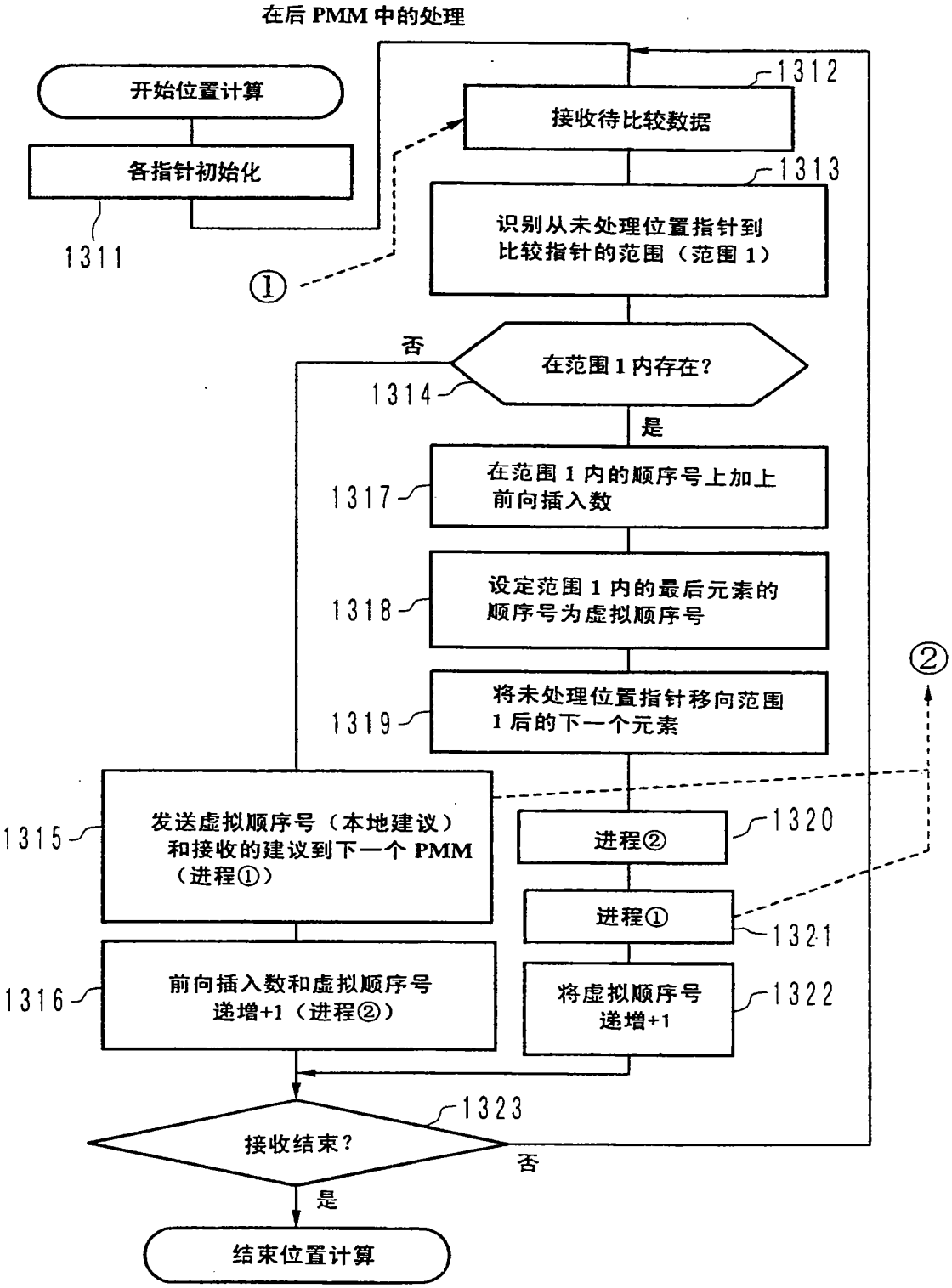

在后 PMM 中的处理
开始位置计算
各指针初始化
1311
1312
接收待比较数据
1313
识别从未处理位置指针到
比较指针的范围（范围 1）
①
否
在范围 1 内存在？
1314
是
1317
在范围 1 内的顺序号上加上
前向插入数
1318
设定范围 1 内的最后元素的
顺序号为虚拟顺序号
1319
将未处理位置指针移向范围
1 后的下一个元素
②
1315
发送虚拟顺序号（本地建议）
和接收的建议到下一个 PMM
（进程①）
进程②
1320
进程①
1321
1316
前向插入数和虚拟顺序号
递增+1（进程②）
将虚拟顺序号
递增+1
1322
1323
接收结束？
否
是
结束位置计算

图 13C

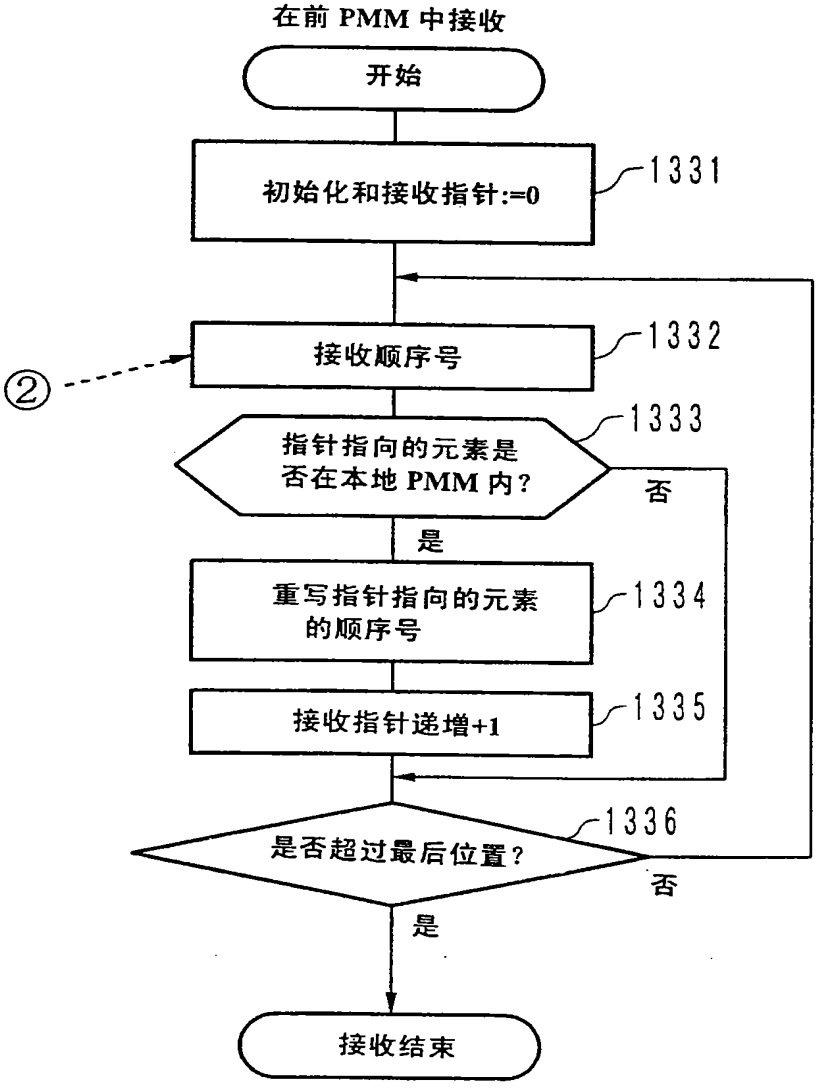

在前 PMM 中接收
开始
初始化和接收指针:=0
1331
接收顺序号
1332
②
指针指向的元素是
否在本地 PMM 内?
1333
否
是
重写指针指向的元素
的顺序号
1334
接收指针递增+1
1335
是否超过最后位置?
1336
否
是
接收结束

图 14A

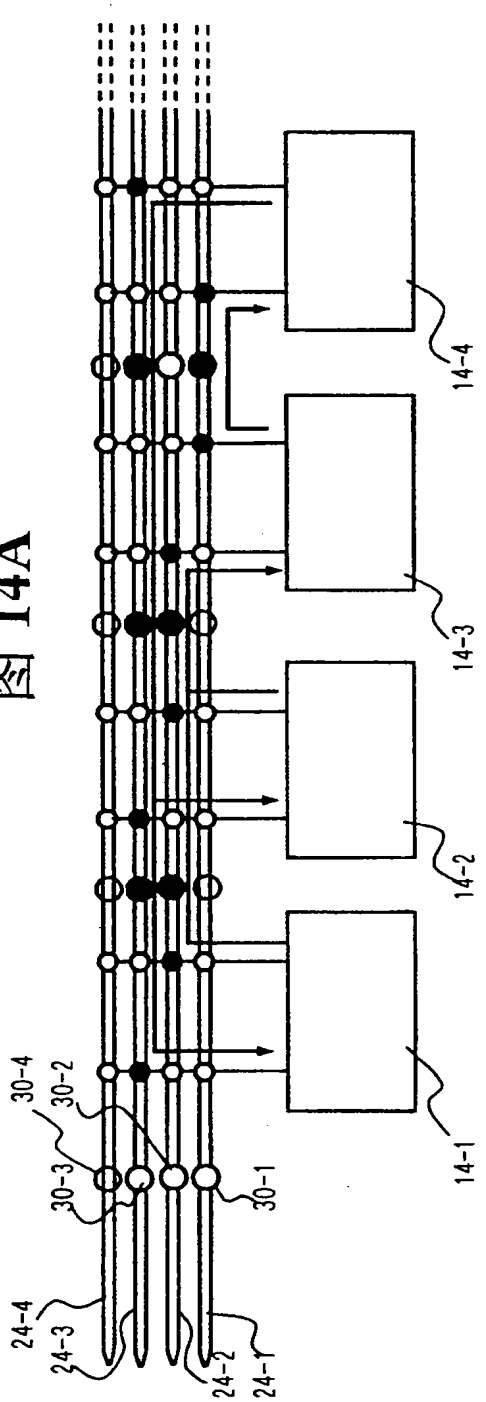

14-4
14-3
14-2
14-1
30-4
30-2
30-3
30-1
24-4
24-3
24-2
24-1

图 14B

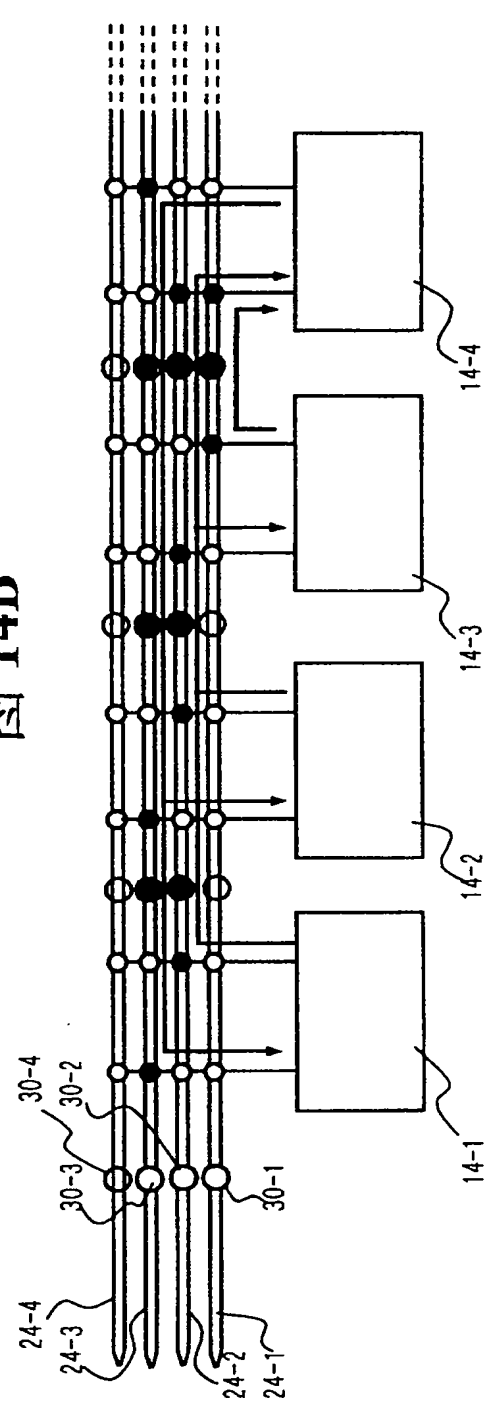

14-4
14-3
14-2
14-1
30-4
30-2
30-3
30-1
24-4
24-3
24-2
24-1

图 15A

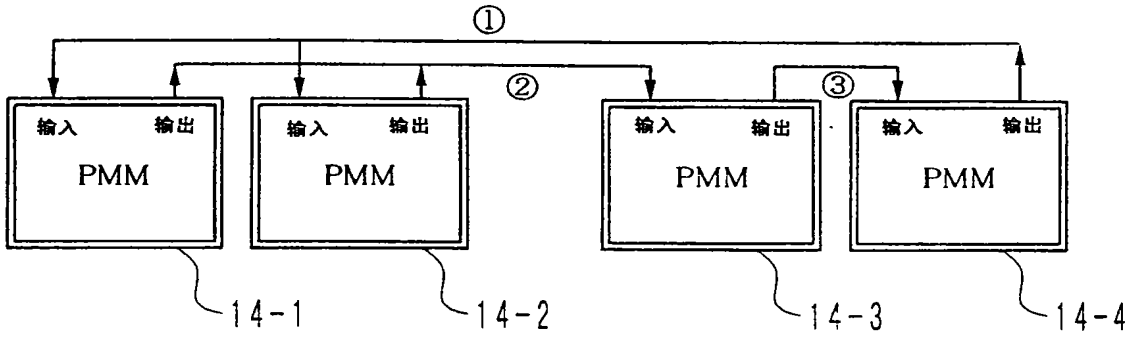


图 15B

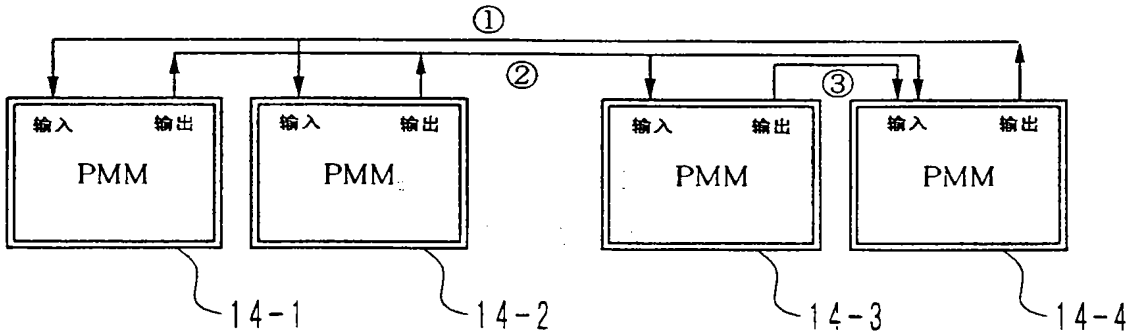


图 16A

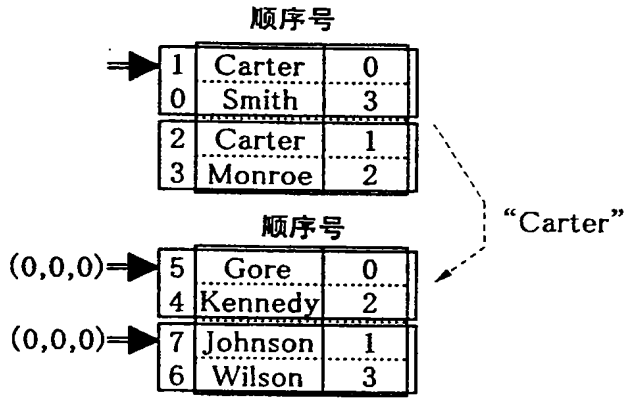

顺序号
1 Carter 0
0 Smith 3
2 Carter 1
3 Monroe 2
"Carter"
顺序号
(0,0,0)
5 Gore 0
4 Kennedy 2
(0,0,0)
7 Johnson 1
6 Wilson 3

图 16B

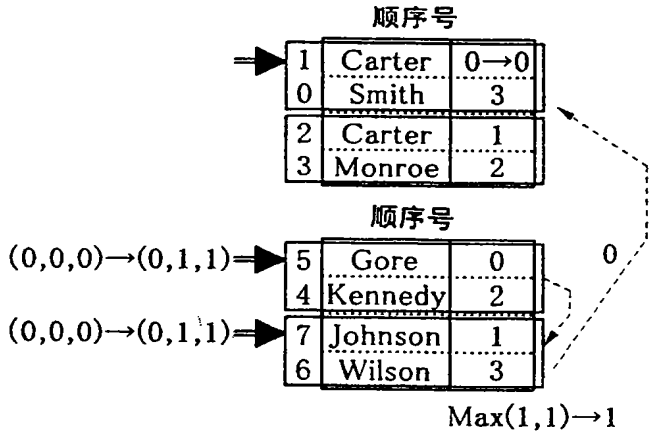

顺序号
1 Carter 0→0
0 Smith 3
2 Carter 1
3 Monroe 2
顺序号
(0,0,0)→(0,1,1)
5 Gore 0
4 Kennedy 2
(0,0,0)→(0,1,1)
7 Johnson 1
6 Wilson 3
0
Max(1,1)→1

图 17A

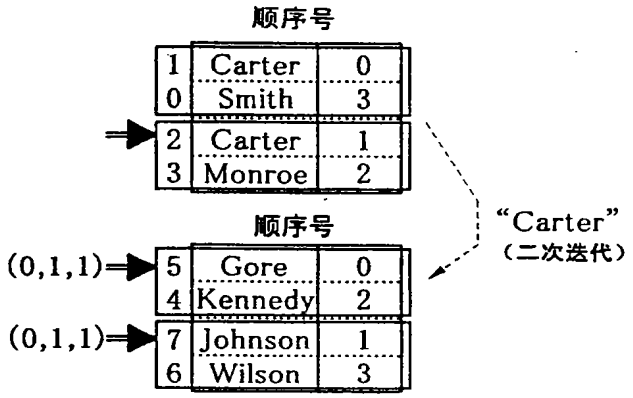

顺序号
1 Carter 0
0 Smith 3
2 Carter 1
3 Monroe 2
顺序号
(0,1,1)
5 Gore 0
4 Kennedy 2
(0,1,1)
7 Johnson 1
6 Wilson 3
"Carter"
(二次迭代)

图 17B

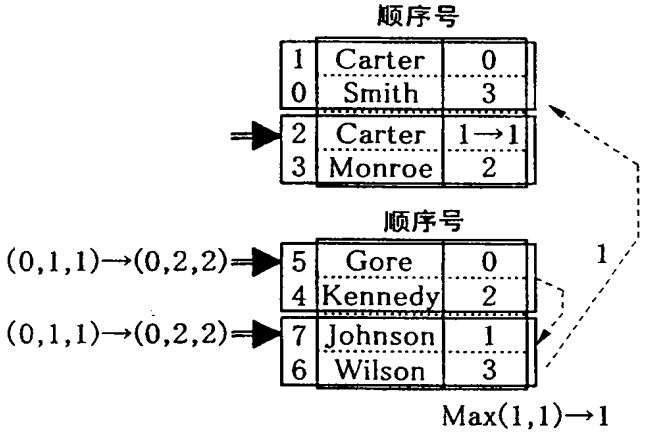

顺序号
1 Carter 0
0 Smith 3
2 Carter 1→1
3 Monroe 2
顺序号
(0,1,1)→(0,2,2)
5 Gore 0
4 Kennedy 2
(0,1,1)→(0,2,2)
7 Johnson 1
6 Wilson 3
1
Max(1,1)→1

图 18A

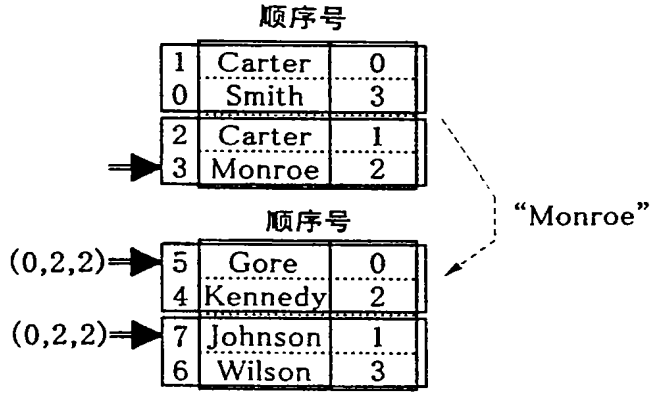

顺序号
1 Carter 0
0 Smith 3
2 Carter 1
3 Monroe 2
"Monroe"
顺序号
(0,2,2)
5 Gore 0
4 Kennedy 2
(0,2,2)
7 Johnson 1
6 Wilson 3

图 18B

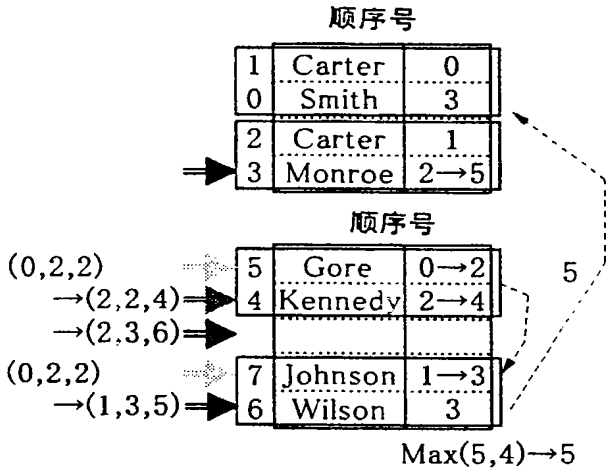

顺序号
1 Carter 0
0 Smith 3
2 Carter 1
3 Monroe 2→5
顺序号
(0,2,2)
→(2,2,4)
→(2,3,6)
5 Gore 0→2
4 Kennedy 2→4
5
(0,2,2)
→(1,3,5)
7 Johnson 1→3
6 Wilson 3
Max(5,4)→5

图 19A

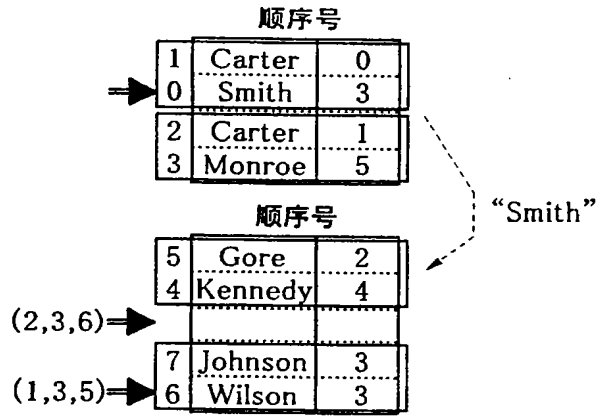

顺序号
1 Carter 0
0 Smith 3
2 Carter 1
3 Monroe 5
"Smith"
顺序号
5 Gore 2
4 Kennedy 4
(2,3,6)
7 Johnson 3
(1,3,5)
6 Wilson 3

图 19B

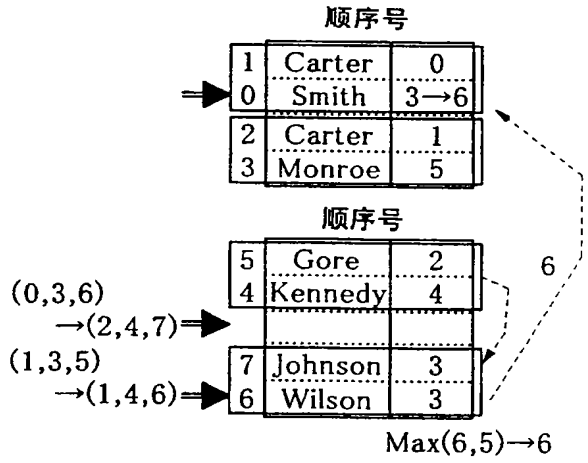

顺序号
1 Carter 0
0 Smith 3→6
2 Carter 1
3 Monroe 5
顺序号
5 Gore 2
4 Kennedy 4
6
(0,3,6)
→(2,4,7)
(1,3,5)
→(1,4,6)
7 Johnson 3
6 Wilson 3
Max(6,5)→6

图 20

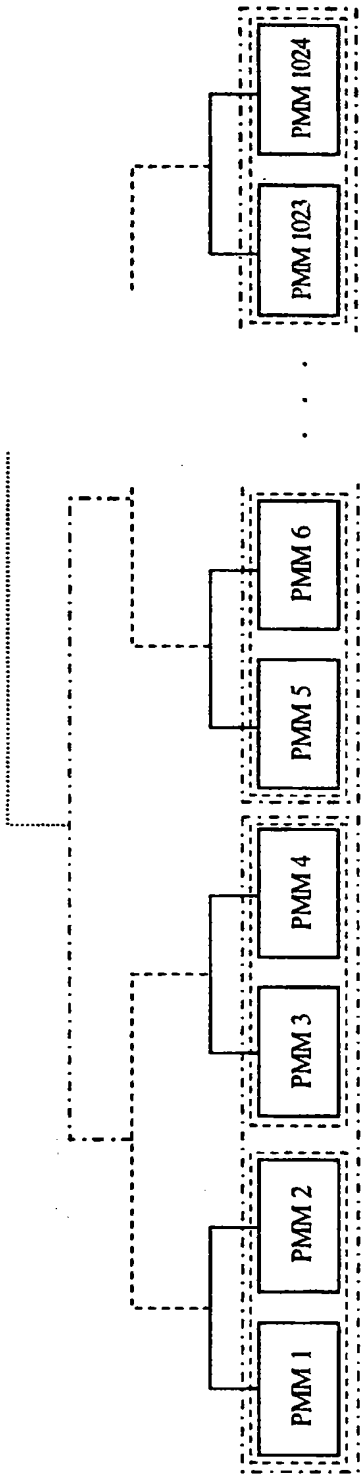


图 21

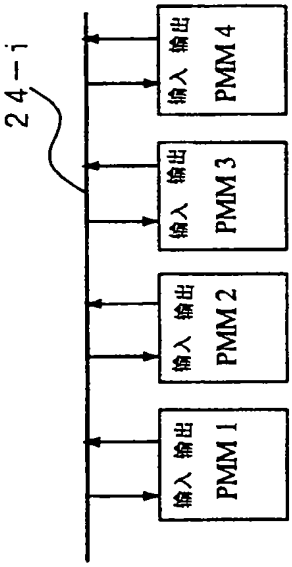

24-i
输入 输出
PMM 4
输入 输出
PMM 3
输入 输出
PMM 2
输入 输出
PMM 1

图 22

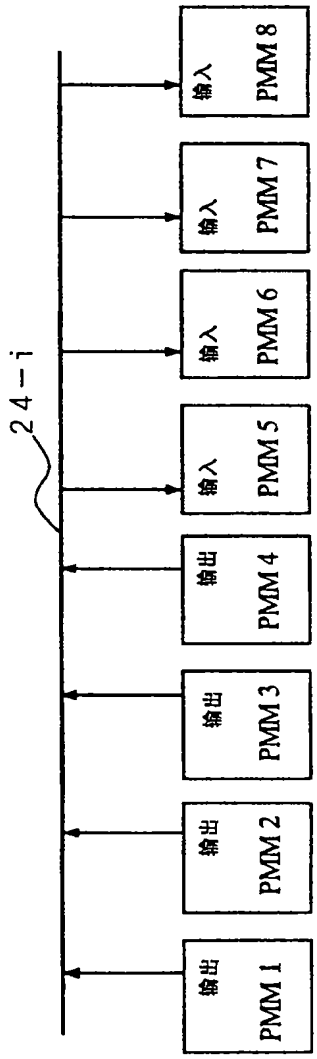

24-i
输入
PMM 8
输入
PMM 7
输入
PMM 6
输入
PMM 5
输出
PMM 4
输出
PMM 3
输出
PMM 2
输出
PMM 1

图 23

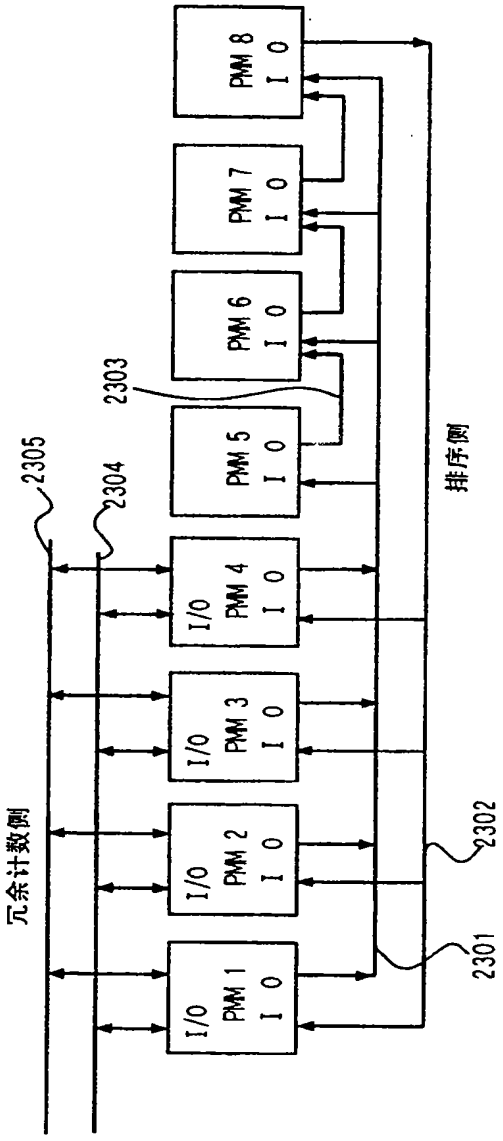

PMM 8
I O
PMM 7
I O
PMM 6
I O
PMM 5
I O
I/O
PMM 4
I O
I/O
PMM 3
I O
I/O
PMM 2
I O
I/O
PMM 1
I O
2303
2305
2304
2302
2301
冗余计数侧
排序侧

图 24

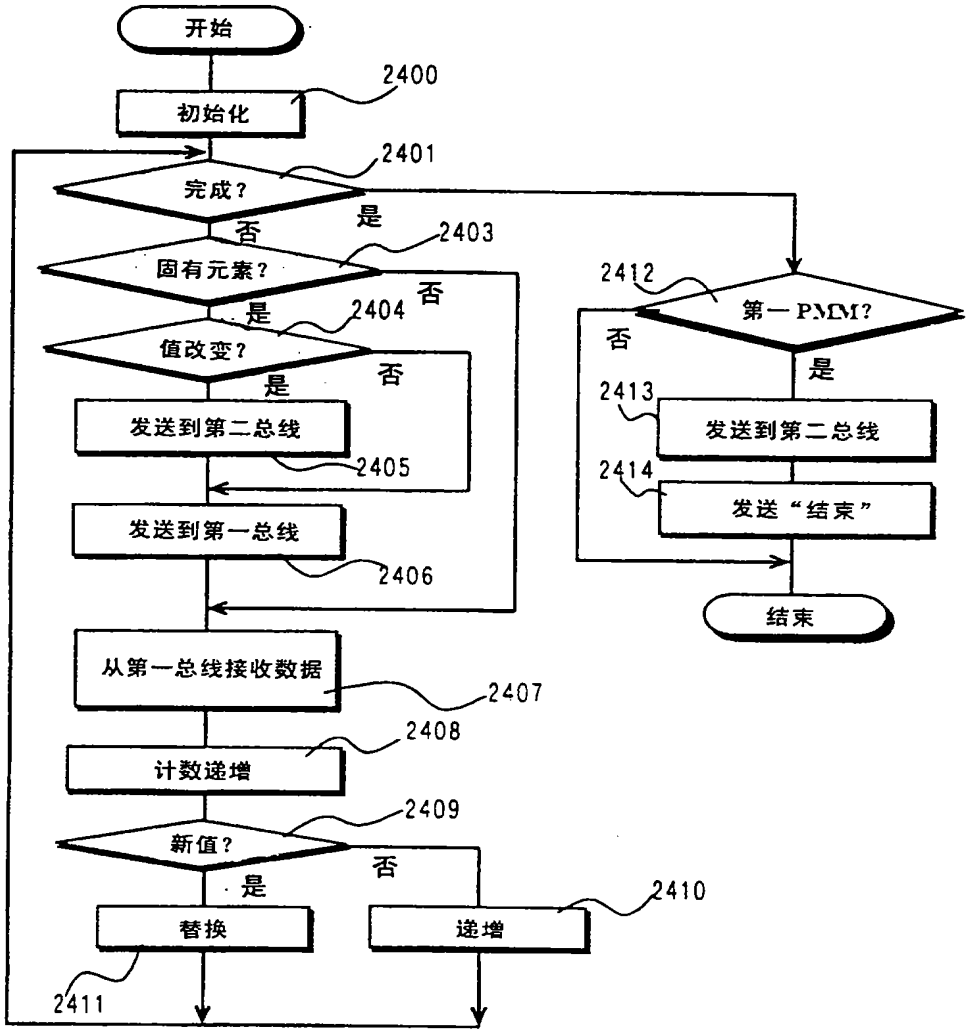

开始
2400
初始化
2401
完成？
是
否
2403
固有元素？
否
是
2404
值改变？
否
是
发送到第二总线
2405
发送到第一总线
2406
从第一总线接收数据
2407
2408
计数递增
2409
新值？
否
是
2410
替换
递增
2411
2412
第一 PMM？
否
是
2413
发送到第二总线
2414
发送“结束”
结束

图 25

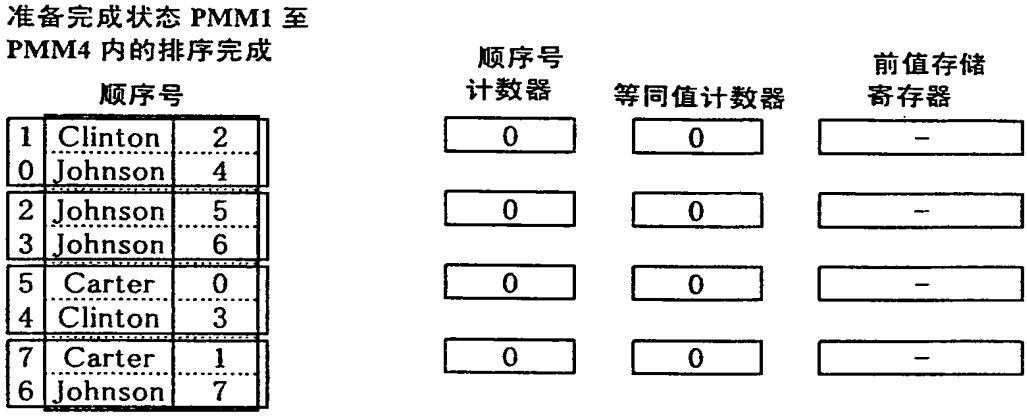

准备完成状态 PMM1 至
PMM4 内的排序完成
顺序号
1 Clinton 2
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton 3
7 Carter 1
6 Johnson 7
顺序号
计数器
等同值计数器
前值存储
寄存器
0
0
-
0
0
-
0
0
-
0
0
-

图 26

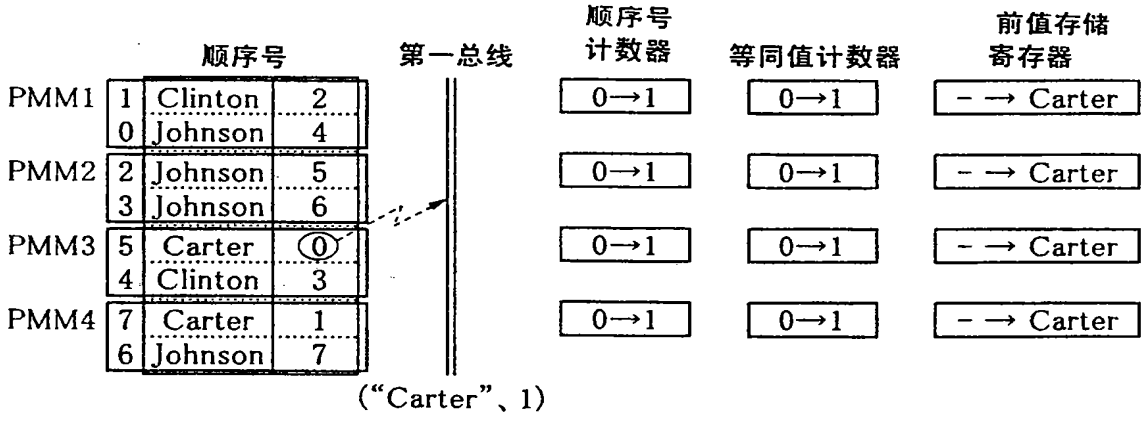

顺序号
第一总线
顺序号
计数器
等同值计数器
前值存储
寄存器
PMM1
1 Clinton 2
0 Johnson 4
PMM2
2 Johnson 5
3 Johnson 6
PMM3
5 Carter 0
4 Clinton 3
PMM4
7 Carter 1
6 Johnson 7
0→1
0→1
- → Carter
0→1
0→1
- → Carter
0→1
0→1
- → Carter
0→1
0→1
- → Carter
("Carter"、1)

图 27

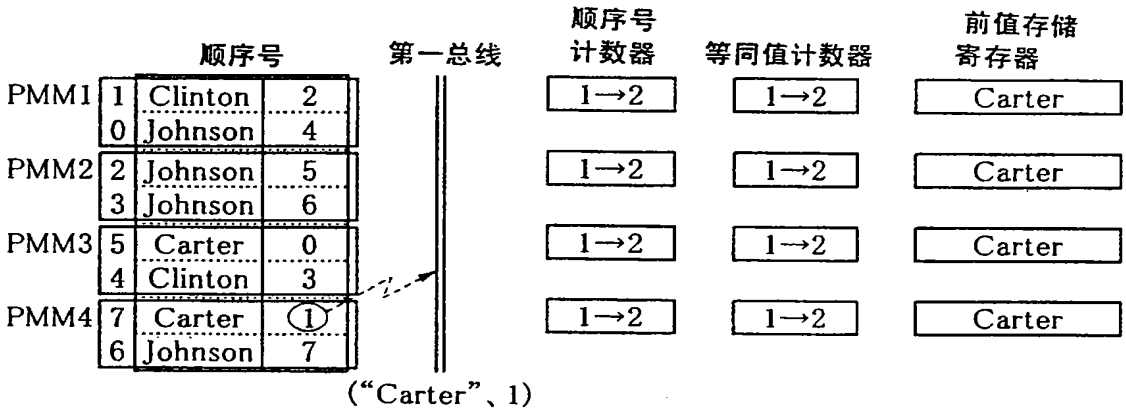

顺序号
第一总线
顺序号计数器
等同值计数器
前值存储寄存器
PMM1
1 Clinton 2
0 Johnson 4
PMM2
2 Johnson 5
3 Johnson 6
PMM3
5 Carter 0
4 Clinton 3
PMM4
7 Carter 1
6 Johnson 7
1→2
1→2
Carter
("Carter"、1)

图 28A

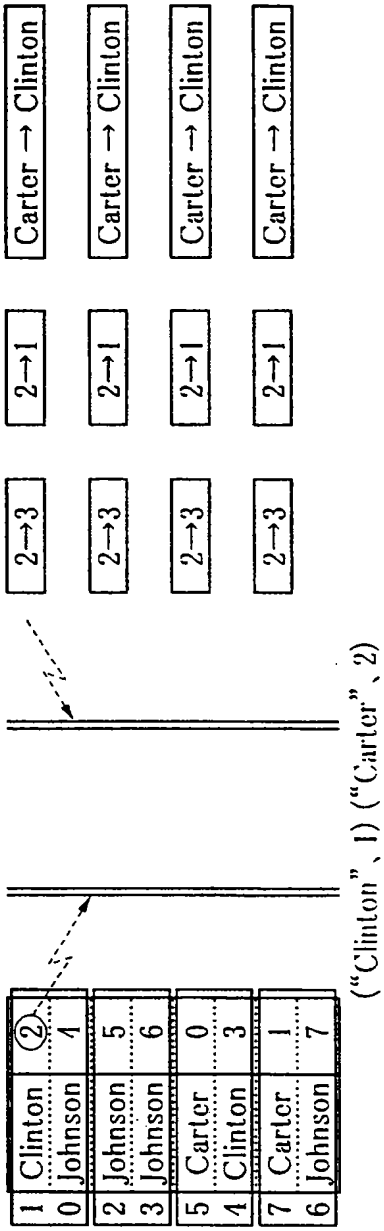

顺序号
第一总线
第二总线
顺序号计数器
等同值计数器
前值存储寄存器
PMM1
PMM2
PMM3
PMM4
1 Clinton ②
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton 3
7 Carter 1
6 Johnson 7
2→3
2→1
Carter → Clinton
("Clinton"、1) ("Carter"、2)

图 28B

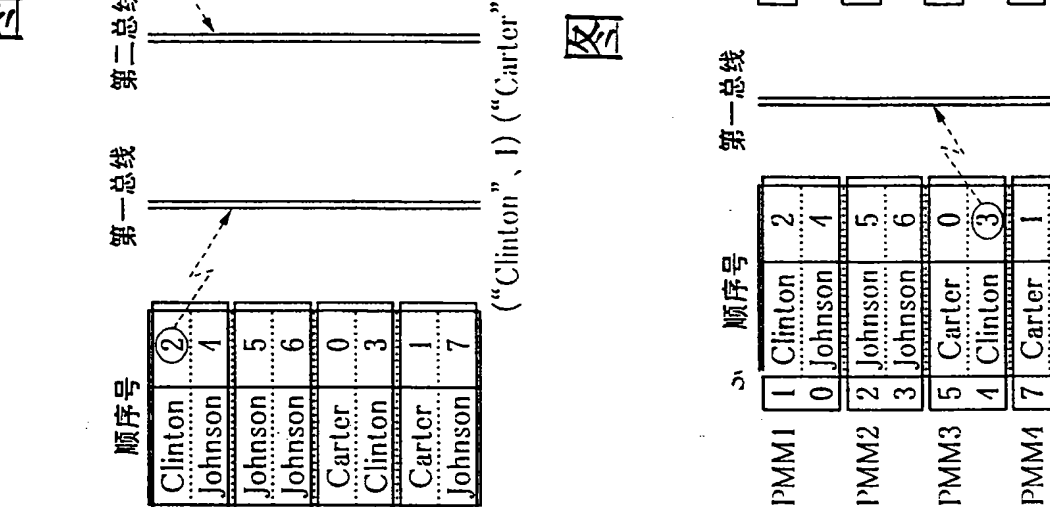

顺序号
第一总线
顺序号计数器
等同值计数器
前值存储寄存器
PMM1
PMM2
PMM3
PMM4
1 Clinton 2
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton ③
7 Carter 1
6 Johnson 7
3→4
1→2
Clinton
("Clinton"、1)

图 29A

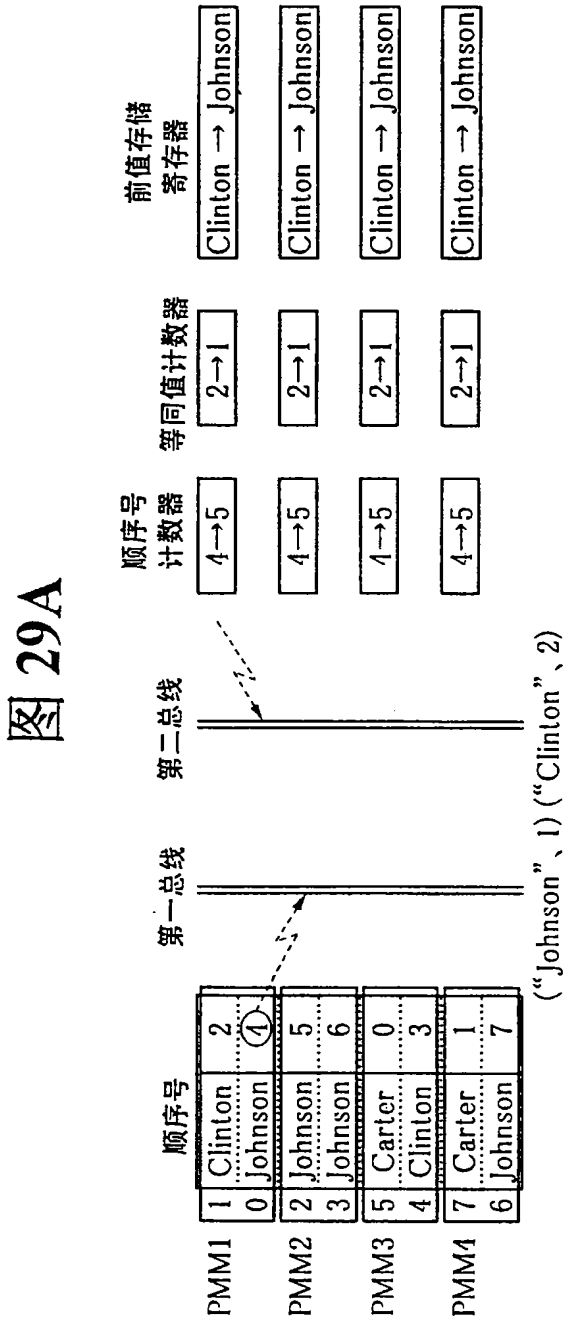

前值存储
寄存器
等同值计数器
顺序号
计数器
第二总线
第一总线
顺序号
PMM1
PMM2
PMM3
PMM4
1 Clinton 2
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton 3
7 Carter 1
6 Johnson 7
4→5
2→1
Clinton → Johnson
("Johnson"、1) ("Clinton"、2)

图 29B

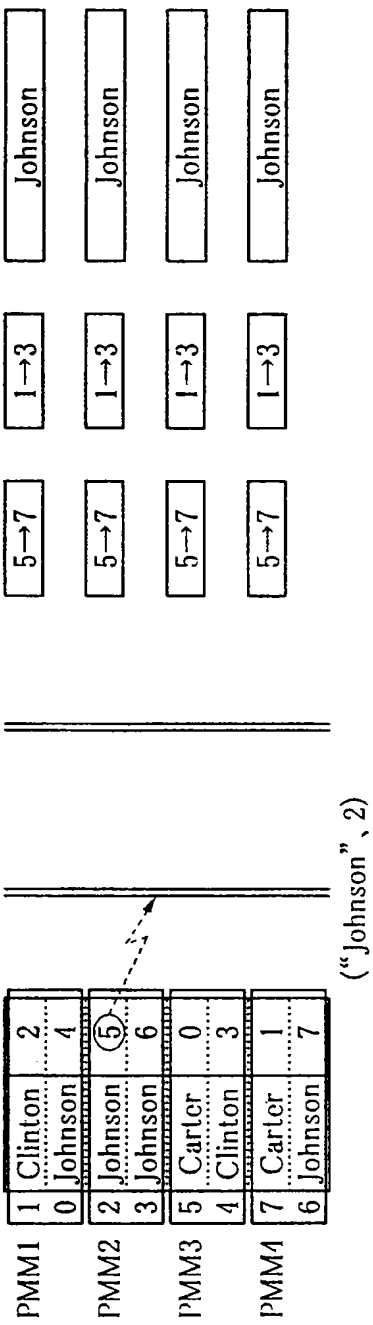

前值存储
寄存器
等同值计数器
顺序号
计数器
第二总线
第一总线
顺序号
PMM1
PMM2
PMM3
PMM4
1 Clinton 2
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton 3
7 Carter 1
6 Johnson 7
5→7
1→3
Johnson
("Johnson"、2)

图 30A

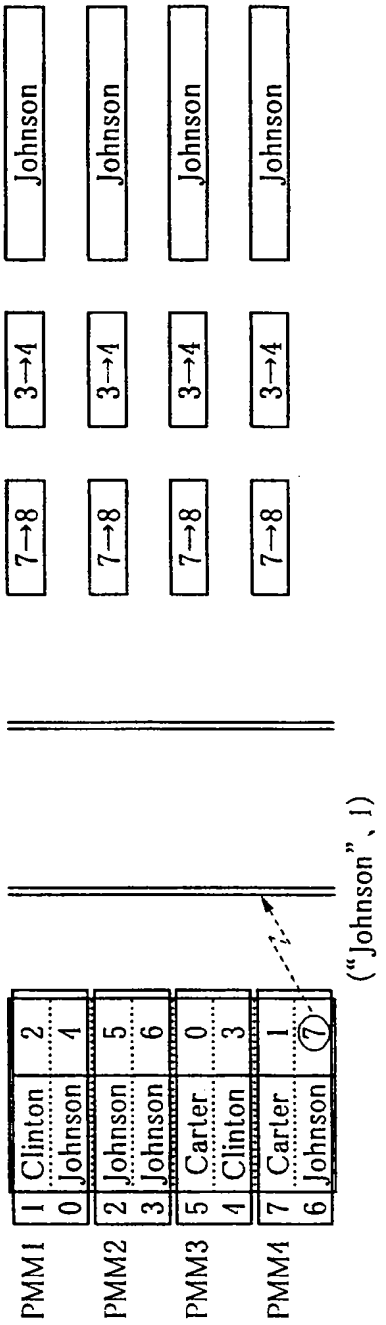

顺序号
第一总线
第二总线
顺序号计数器
等同值计数器
前值存储寄存器
PMM1
PMM2
PMM3
PMM4
1 Clinton 2
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton 3
7 Carter 1
6 Johnson 7
7→8
3→4
Johnson
("Johnson"、1)

图 30B

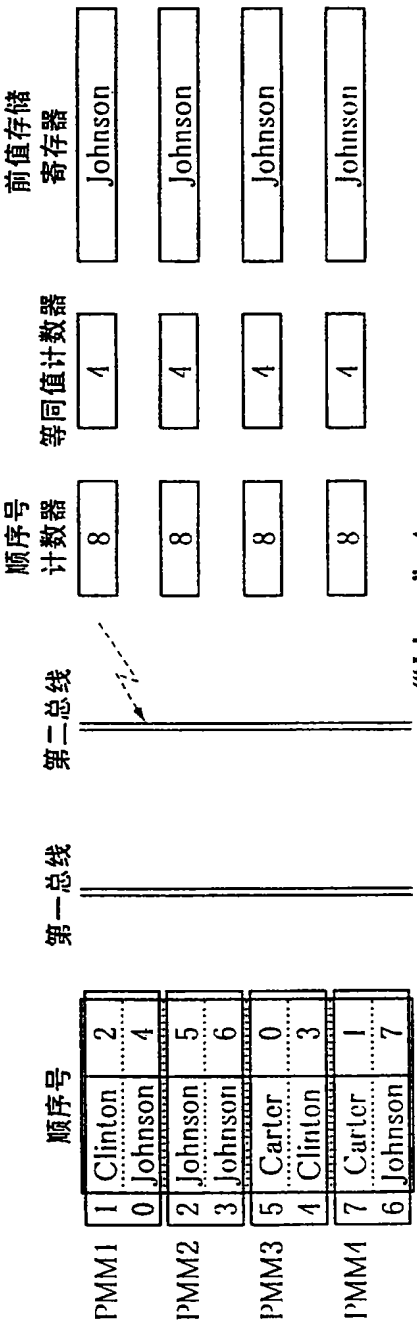

顺序号
第一总线
第二总线
顺序号计数器
等同值计数器
前值存储寄存器
PMM1
PMM2
PMM3
PMM4
1 Clinton 2
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton 3
7 Carter 1
6 Johnson 7
8
1
Johnson
("Johnson"、4
"结束")

图 31A

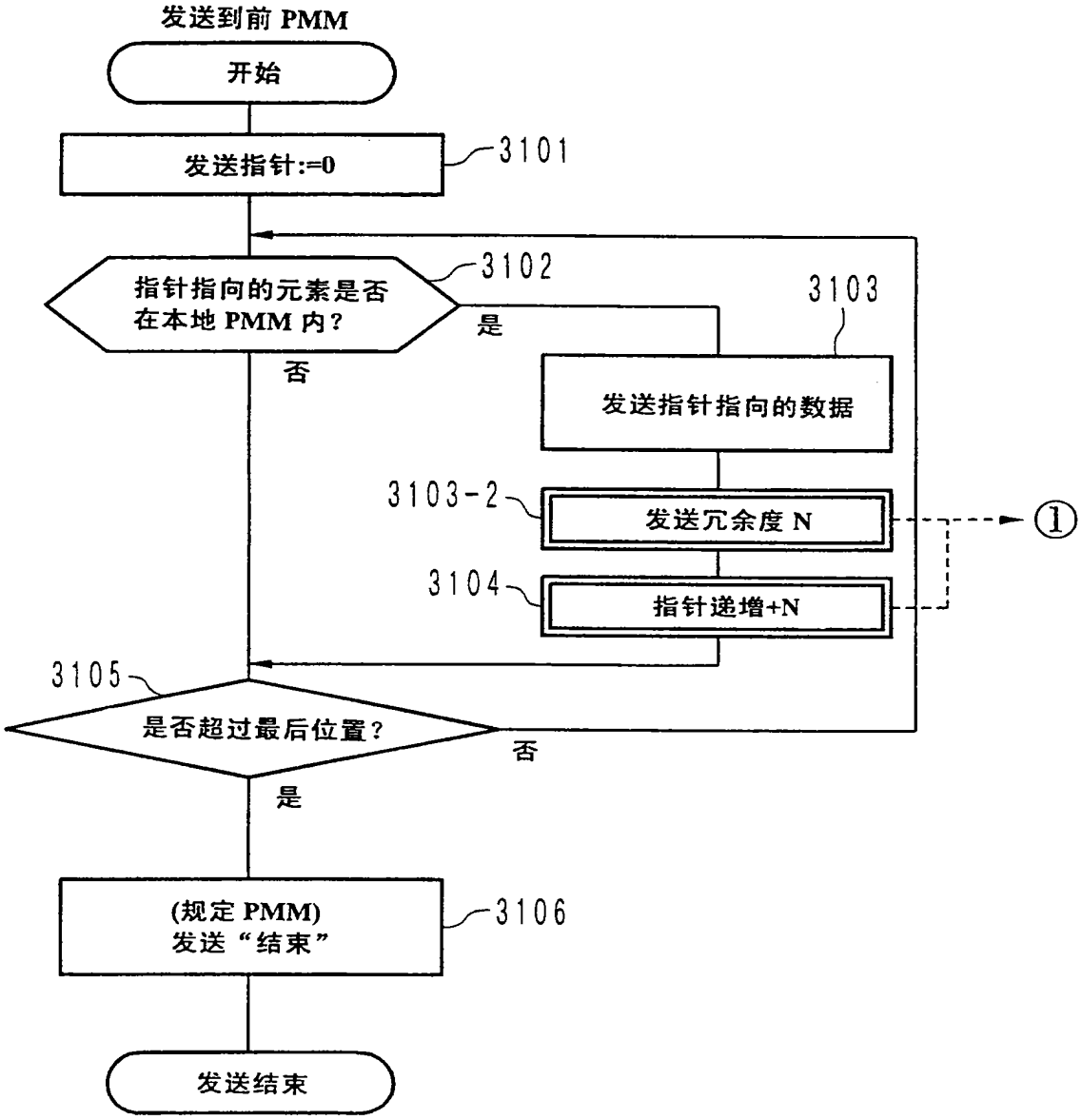

发送到前 PMM
开始
发送指针:=0
3101
指针指向的元素是否
在本地 PMM 内?
3102
是
否
3103
发送指针指向的数据
3103-2
发送冗余度 N
①
3104
指针递增+N
3105
是否超过最后位置?
否
是
(规定 PMM)
发送“结束”
3106
发送结束

图 31B

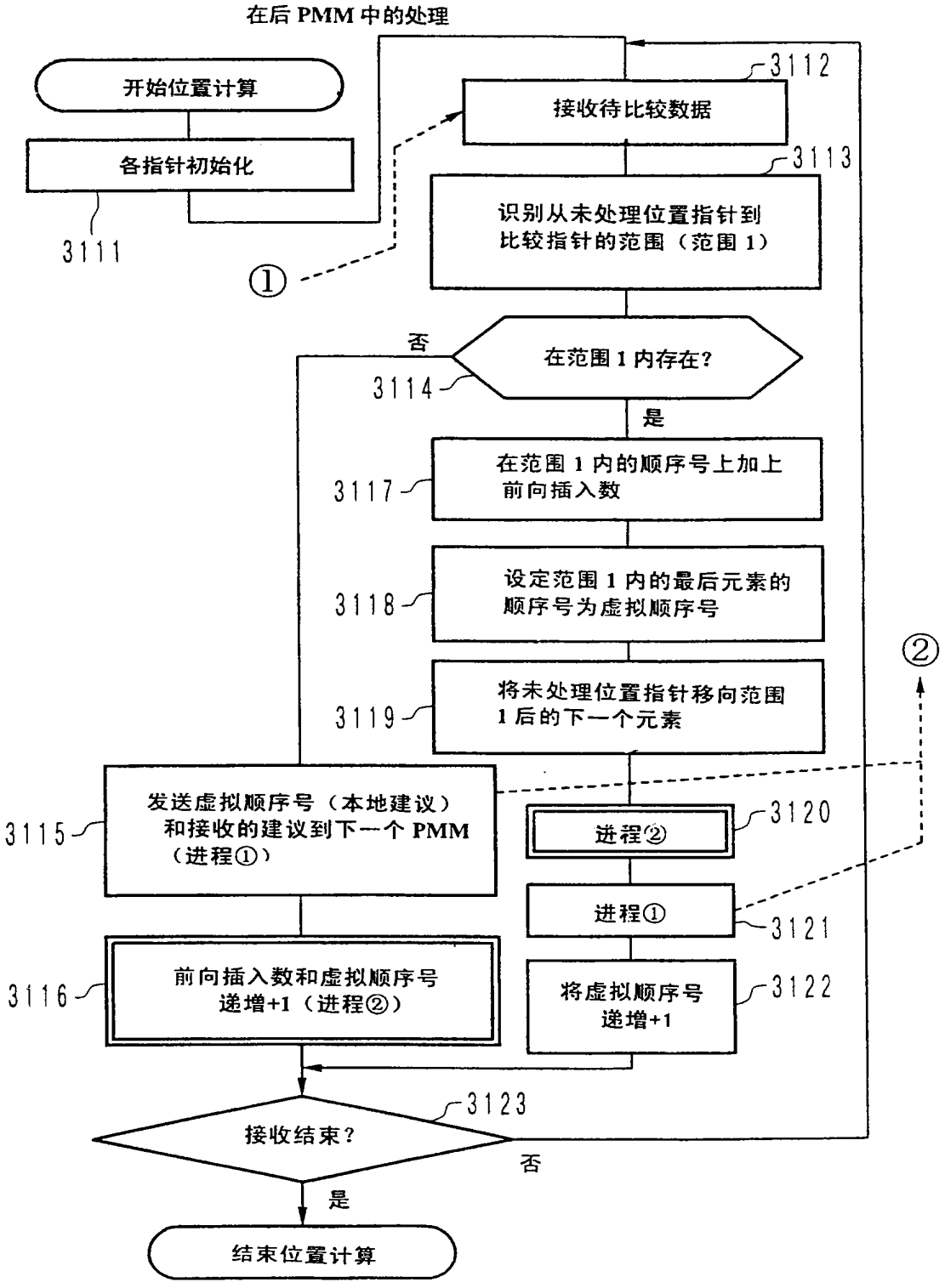

在后 PMM 中的处理
开始位置计算
各指针初始化
3111
3112
接收待比较数据
3113
识别从未处理位置指针到
比较指针的范围（范围 1）
①
否
在范围 1 内存在?
3114
是
3117
在范围 1 内的顺序号上加上
前向插入数
3118
设定范围 1 内的最后元素的
顺序号为虚拟顺序号
3119
将未处理位置指针移向范围
1 后的下一个元素
②
3115
发送虚拟顺序号（本地建议）
和接收的建议到下一个 PMM
（进程①）
进程②
3120
进程①
3121
3116
前向插入数和虚拟顺序号
递增+1（进程②）
将虚拟顺序号
递增+1
3122
3123
接收结束?
否
是
结束位置计算

图 31C

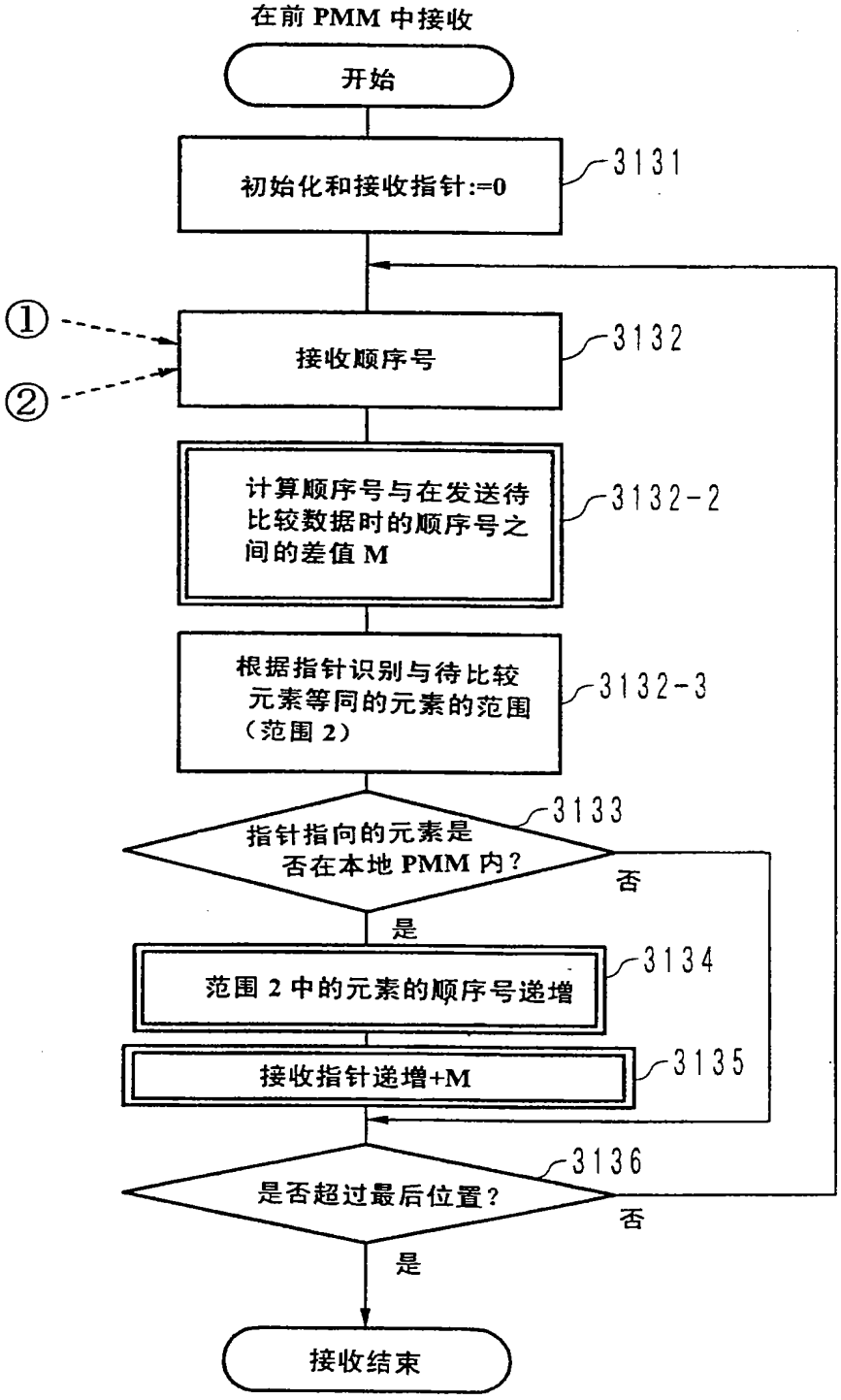

在前 PMM 中接收
开始
初始化和接收指针:=0
3131
①
②
接收顺序号
3132
计算顺序号与在发送待比较数据时的顺序号之间的差值 M
3132-2
根据指针识别与待比较元素等同的元素的范围（范围 2）
3132-3
指针指向的元素是否在本地 PMM 内？
3133
否
是
范围 2 中的元素的顺序号递增
3134
接收指针递增+M
3135
是否超过最后位置？
3136
否
是
接收结束

图 32

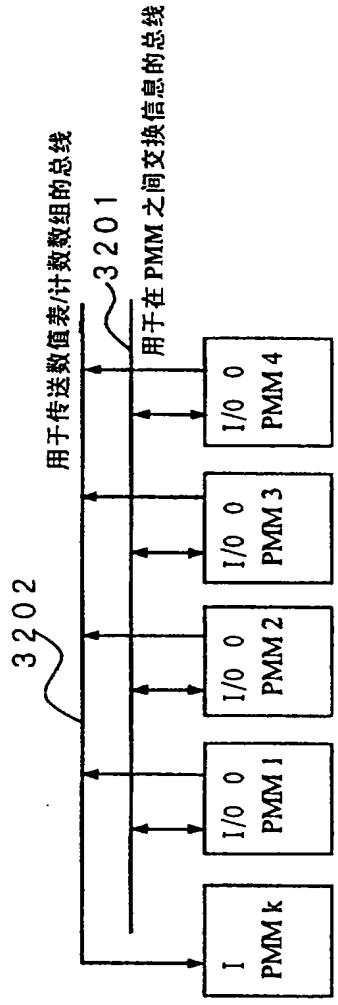

用于传送数值表/计数数组的总线
3201
用于在 PMM 之间交换信息的总线
3202
I/O
PMM k
I/O
PMM 1
I/O
PMM 2
I/O
PMM 3
I/O
PMM 4

图 33

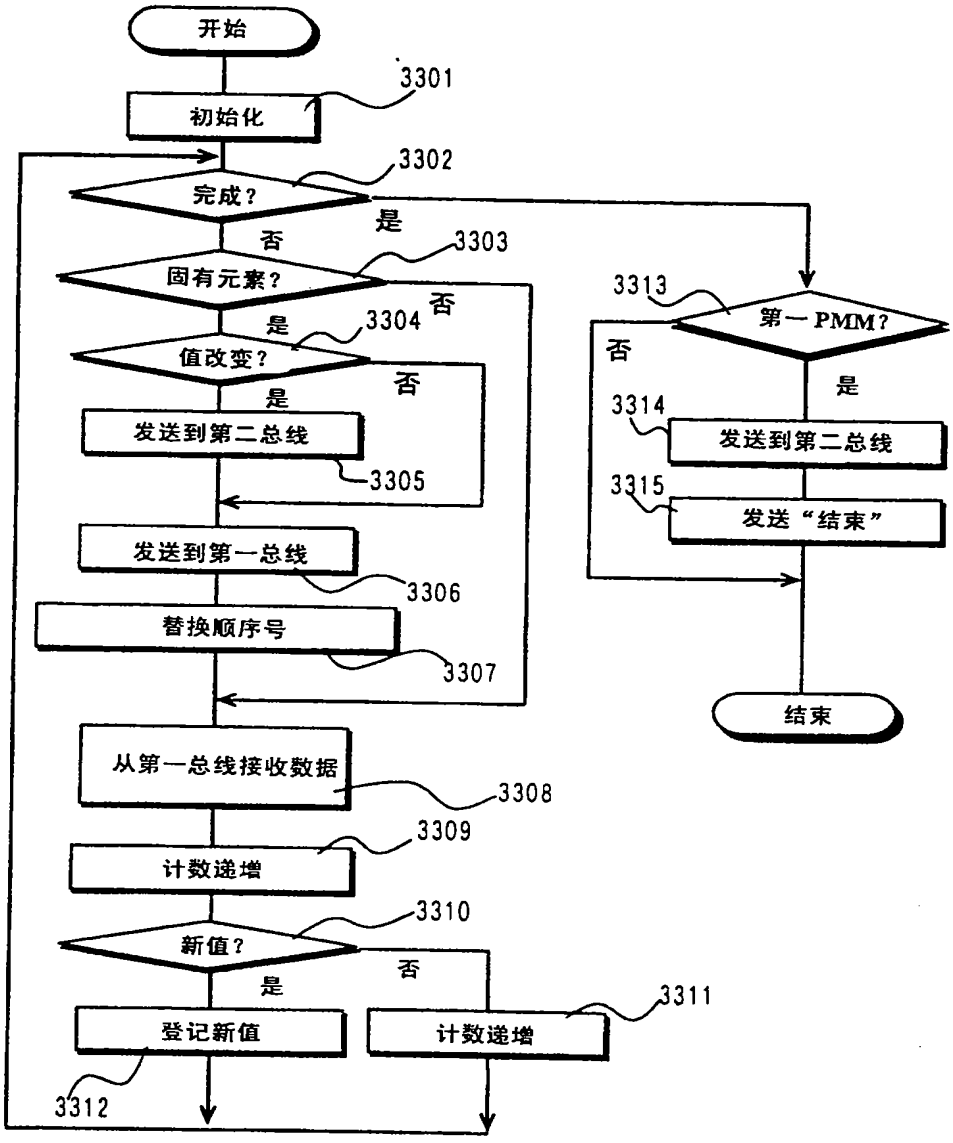

开始
3301
初始化
3302
完成？
是
否
3303
固有元素？
否
是
3304
值改变？
否
是
发送到第二总线
3305
发送到第一总线
3306
替换顺序号
3307
从第一总线接收数据
3308
3309
计数递增
3310
新值？
否
是
登记新值
3311
计数递增
3312
3313
第一 PMM？
否
是
3314
发送到第二总线
3315
发送“结束”
结束

图 34A

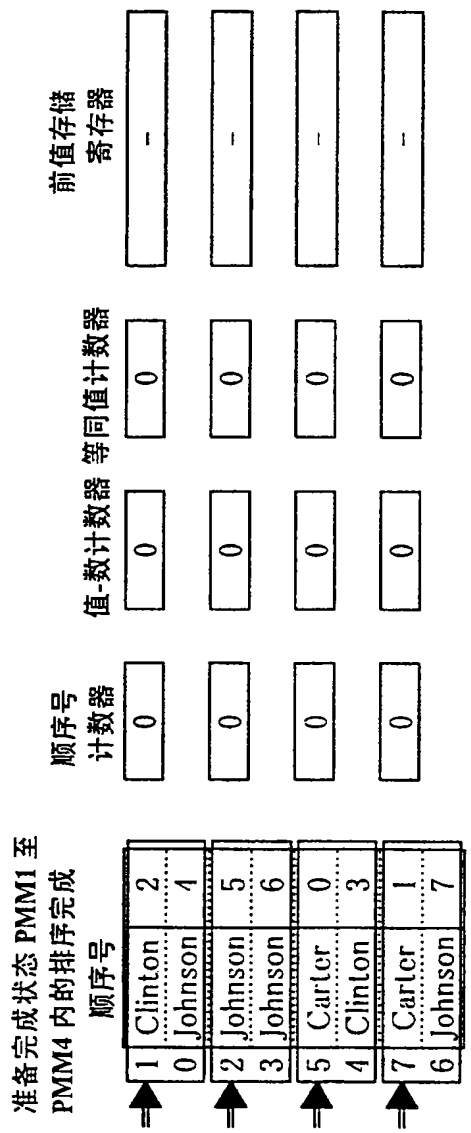

准备完成状态 PMM1 至 PMM4 内的排序完成
顺序号
顺序号 计数器
值-数计数器
等同值计数器
前值存储 寄存器
1 Clinton 2
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton 3
7 Carter 1
6 Johnson 7
0
0
0
0
0
0
0
0
-
-
-
-

图 34B

顺序号
顺序号 计数器
值-数计数器
等同值计数器
前值存储 寄存器
第一总线
PMM1
PMM2
PMM3
PMM4
1 Clinton 2
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0→0
4 Clinton 3
7 Carter 1
6 Johnson 7
0→1
0→1
0→1
0→1
0
0
0
0
- → Carter
- → Carter
- → Carter
- → Carter
("Carter"、1)

图 35A

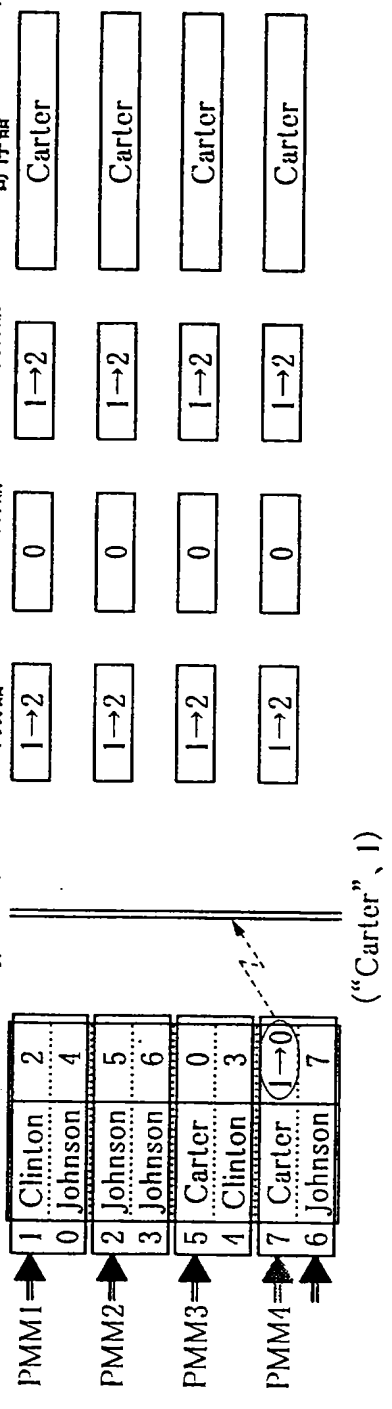

顺序号
计数器
值-数计数器
等同值计数器
前值存储
寄存器
1→2
0
Carter
PMM1
PMM2
PMM3
PMM4
1 Clinton 2
0 Johnson 4
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton 3
7 Carter (1→0)
6 Johnson 7
第一总线
("Carter"、1)

图 35B

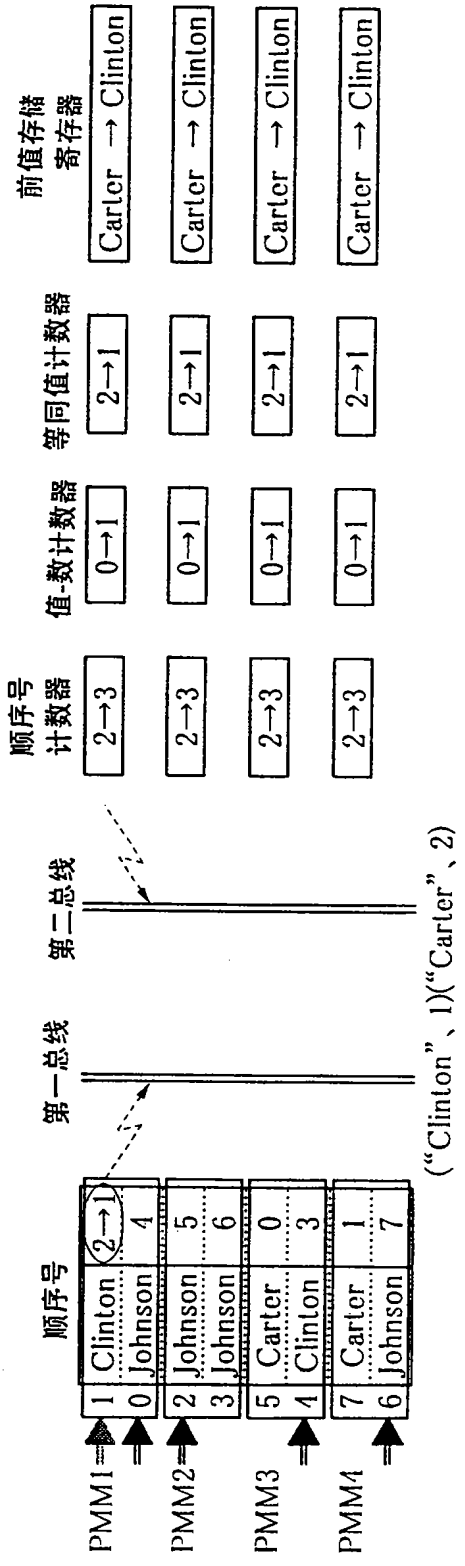

顺序号
计数器
值-数计数器
等同值计数器
前值存储
寄存器
2→3
0→1
2→1
Carter → Clinton
PMM1
PMM2
PMM3
PMM4
1 Clinton (2→1) 4
0 Johnson
2 Johnson 5
3 Johnson 6
5 Carter 0
4 Clinton 3
7 Carter 1
6 Johnson 7
第一总线
第二总线
("Clinton"、1)("Carter"、2)

图 36A

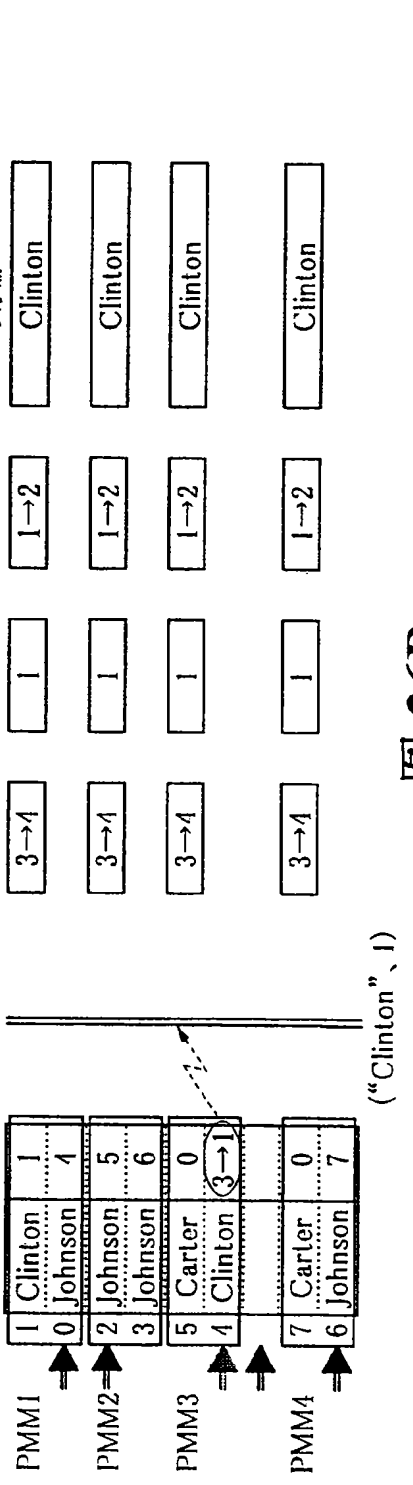

顺序号
第一总线
顺序号 计数器
值-数计数器
等同值计数器
前值存储 寄存器
PMM1
PMM2
PMM3
PMM4
1 Clinton
0 Johnson
2 Johnson
3 Johnson
5 Carter
4 Clinton
7 Carter
6 Johnson
3→4
1→2
Clinton
("Clinton"、1)

图 36B

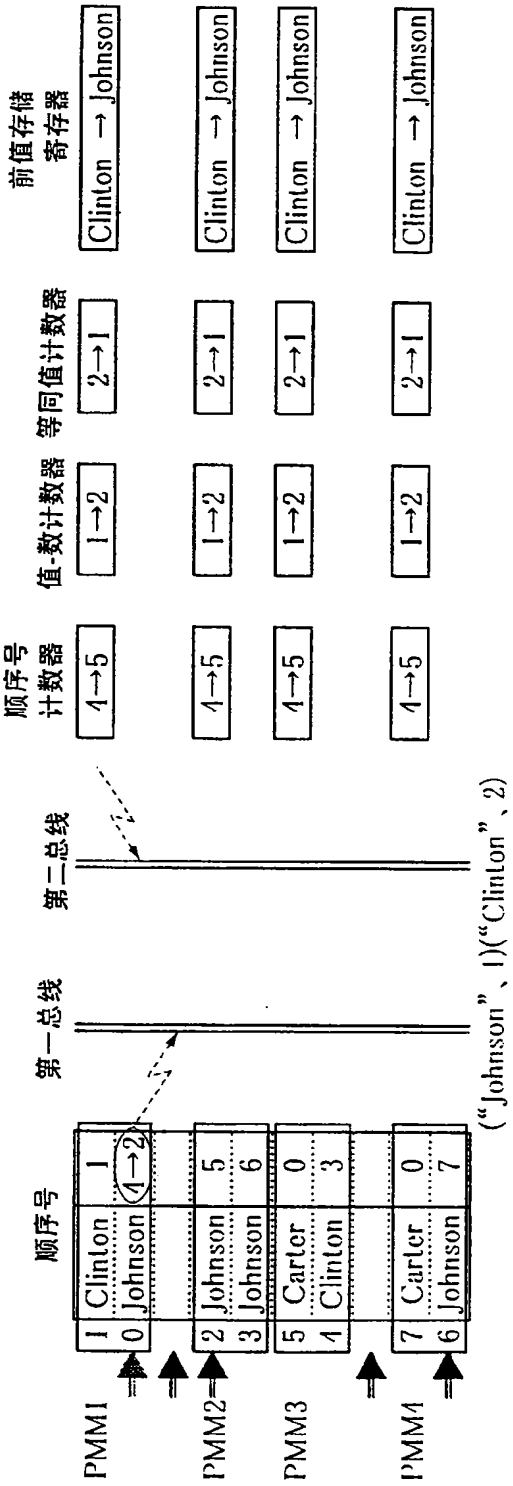

顺序号
第一总线
第二总线
顺序号 计数器
值-数计数器
等同值计数器
前值存储 寄存器
PMM1
PMM2
PMM3
PMM4
1 Clinton
0 Johnson
2 Johnson
3 Johnson
5 Carter
4 Clinton
7 Carter
6 Johnson
1→5
1→2
2→1
Clinton → Johnson
("Johnson"、1)("Clinton"、2)

图 37A

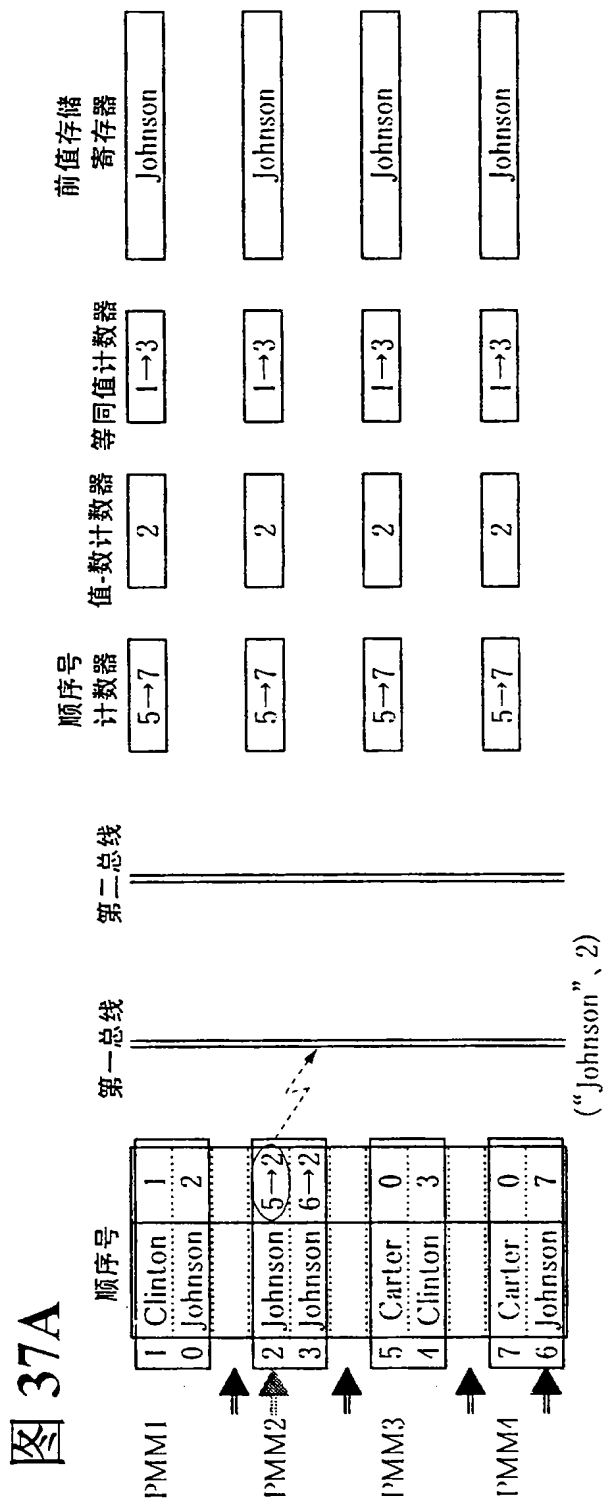

PMM1
PMM2
PMM3
PMM4
顺序号
1 Clinton
0 Johnson
2 Johnson (5→2)
3 Johnson 6→2
5 Carter
4 Clinton
7 Carter
6 Johnson
第一总线
第二总线
("Johnson"、2)
顺序号计数器
5→7
值-数计数器
2
等同值计数器
1→3
前值存储寄存器
Johnson

图 37B

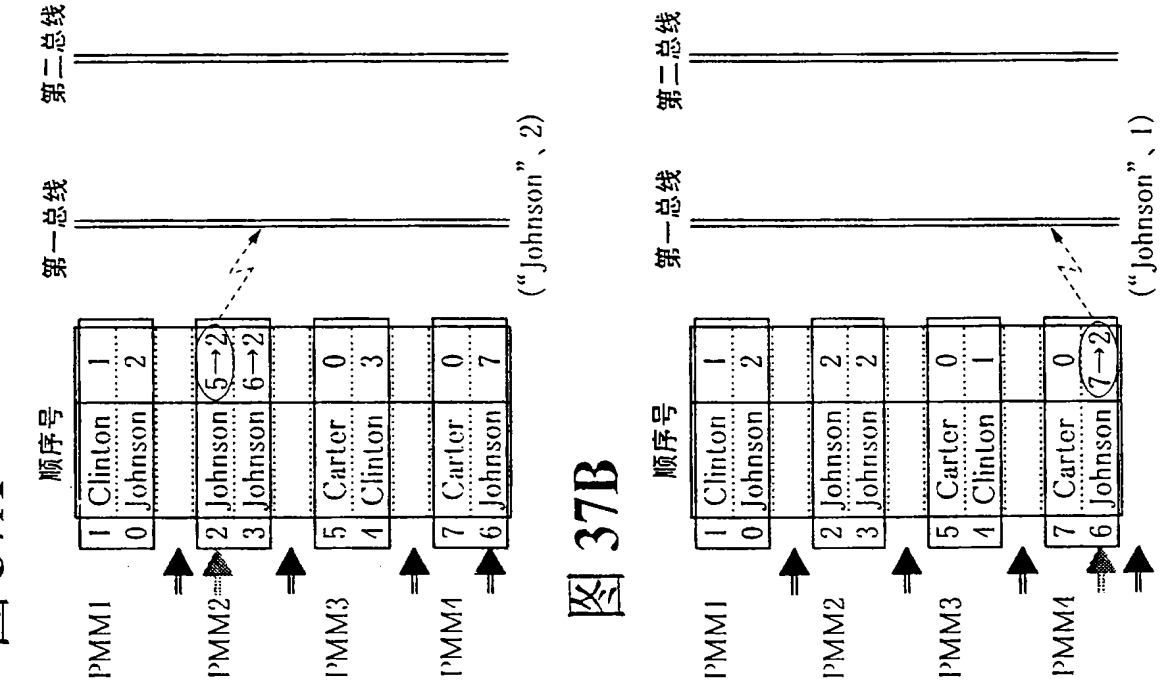

PMM1
PMM2
PMM3
PMM4
顺序号
1 Clinton
0 Johnson
2 Johnson
3 Johnson
5 Carter
4 Clinton
7 Carter
6 Johnson (7→2)
第一总线
第二总线
("Johnson"、1)
顺序号计数器
7→8
值-数计数器
2
等同值计数器
3→4
前值存储寄存器
Johnson

图 38

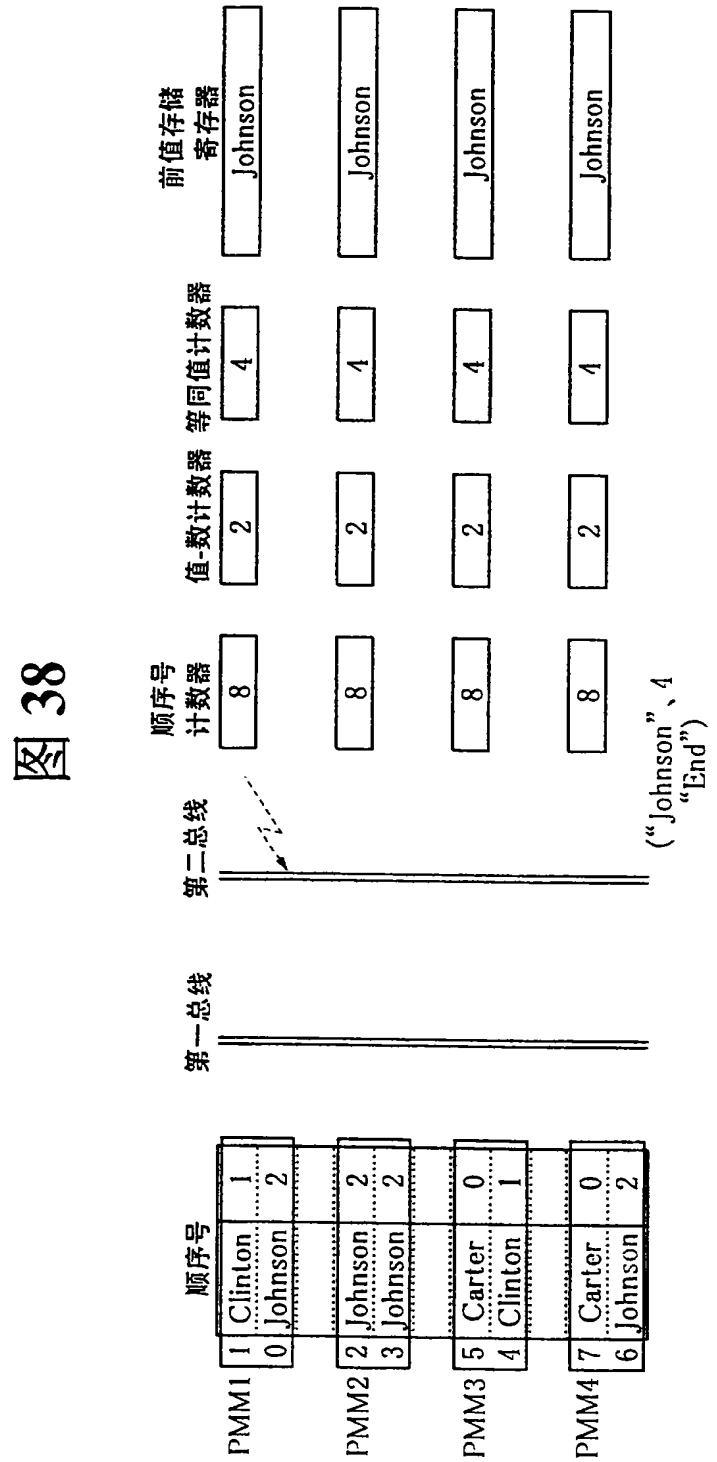

前值存储
寄存器
Johnson
等同值计数器
值-数计数器
顺序号
计数器
第二总线
("Johnson"、4
"End")
第一总线
顺序号
PMM1
1 Clinton 1
0 Johnson 2
PMM2
2 Johnson 2
3 Johnson 2
PMM3
5 Carter 0
4 Clinton 1
PMM4
7 Carter 0
6 Johnson 2

图 39A

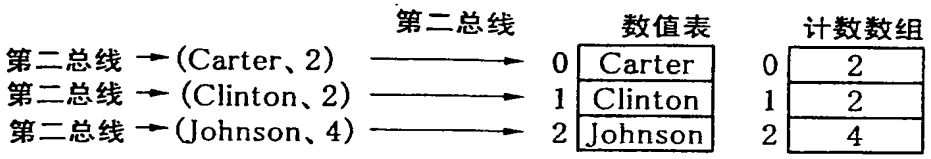

第二总线
数值表
计数数组
第二总线 (Carter、2)
第二总线 (Clinton、2)
第二总线 (Johnson、4)
0 Carter 0 2
1 Clinton 1 2
2 Johnson 2 4

图 39B

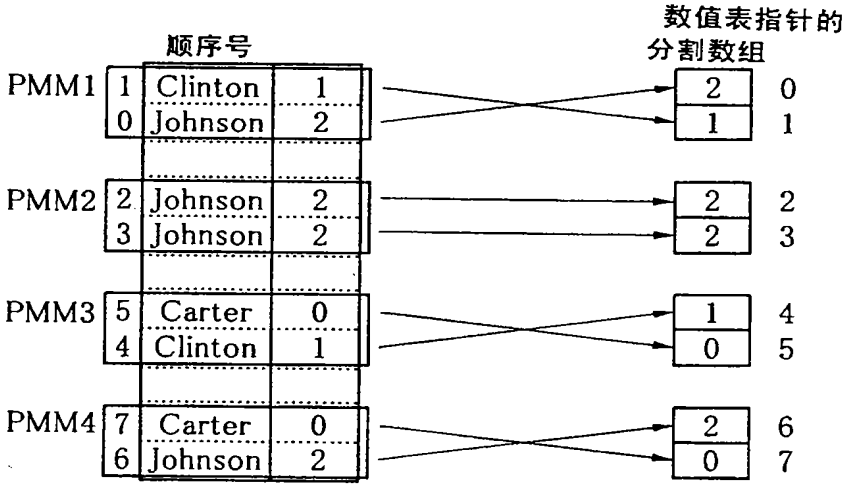

顺序号
数值表指针的
分割数组
PMM1
1 Clinton 1
0 Johnson 2
PMM2
2 Johnson 2
3 Johnson 2
PMM3
5 Carter 0
4 Clinton 1
PMM4
7 Carter 0
6 Johnson 2
2 0
1 1
2 2
2 3
1 4
0 5
2 6
0 7

图 40

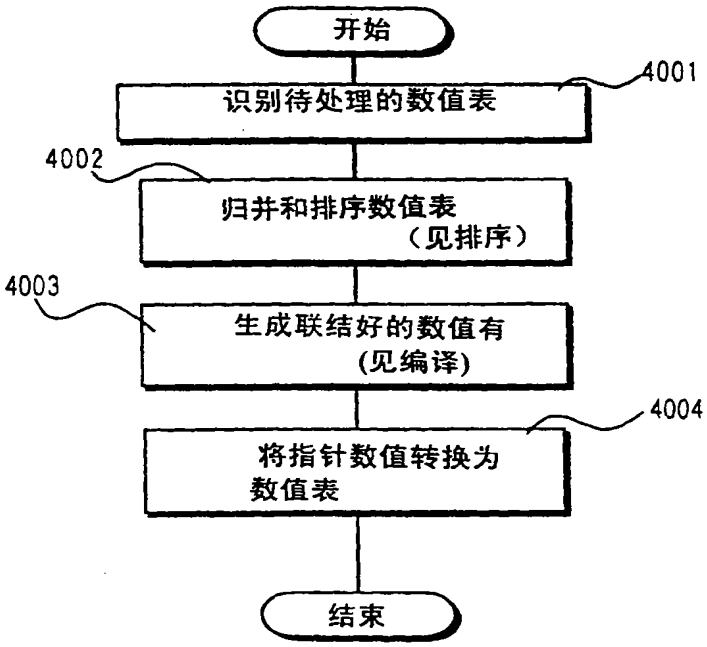

开始
4001
识别待处理的数值表
4002
归并和排序数值表
（见排序）
4003
生成联结好的数值有
(见编译)
4004
将指针数值转换为
数值表
结束

图 41A

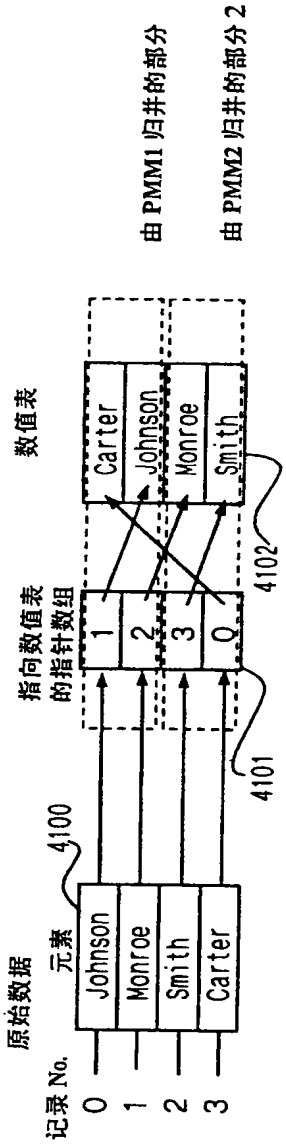

原始数据
元素
记录 No.
Johnson
Monroe
Smith
Carter
0
1
2
3
4100
指向数值表
的指针数组
1
2
3
0
4101
数值表
Carter
Johnson
Monroe
Smith
4102
由 PMM1 归并的部分
由 PMM2 归并的部分 2

图 41B

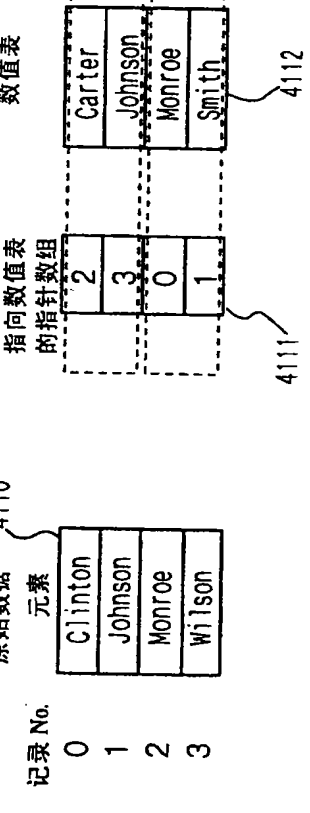
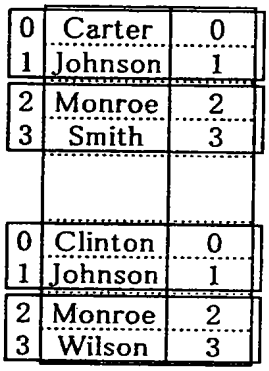

原始数据
元素
记录 No.
Clinton
Johnson
Monroe
Wilson
0
1
2
3
4110
指向数值表
的指针数组
2
3
0
1
4111
数值表
Carter
Johnson
Monroe
Smith
4112
由 PMM3 归并的部分
由 PMM4 归并的部分

图 42A



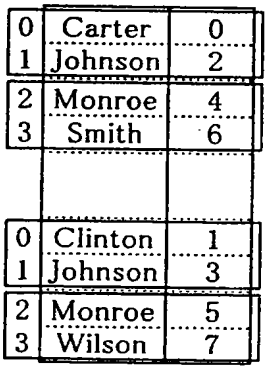
由 PMM1 管理的范围

由 PMM2 管理的范围

由 PMM3 管理的范围(PMM1 也可能)

由 PMM4 管理的范围(PMM1 也可能)

图 42B



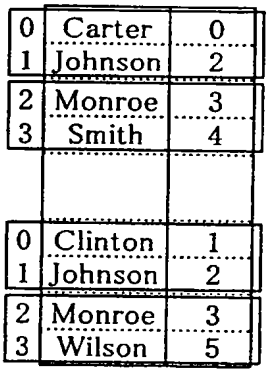
由 PMM1 管理的范围

由 PMM2 管理的范围

由 PMM3 管理的范围(PMM1 也可能)

由 PMM4 管理的范围(PMM1 也可能)

图 42C

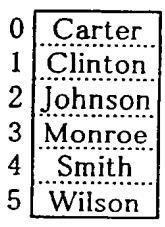


由 PMM1 管理的范围

由 PMM2 管理的范围

由 PMM3 管理的范围(PMM1 也可能)

由 PMM4 管理的范围(PMM1 也可能)



联结好的计数数组

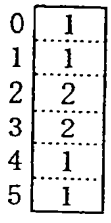


图 43A

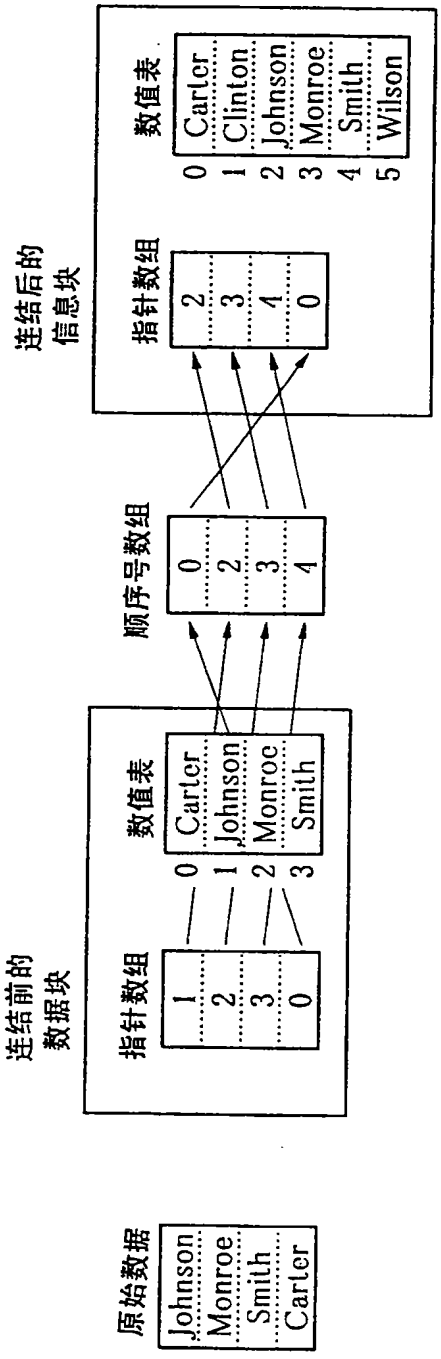

连结后的
信息块
指针数组
2
3
1
0
数值表
0 Carter
1 Clinton
2 Johnson
3 Monroe
4 Smith
5 Wilson
顺序号数组
0
2
3
4
连结前的
数据块
指针数组
1
2
3
0
数值表
0 Carter
1 Johnson
2 Monroe
3 Smith
原始数据
Johnson
Monroe
Smith
Carter

图 43B

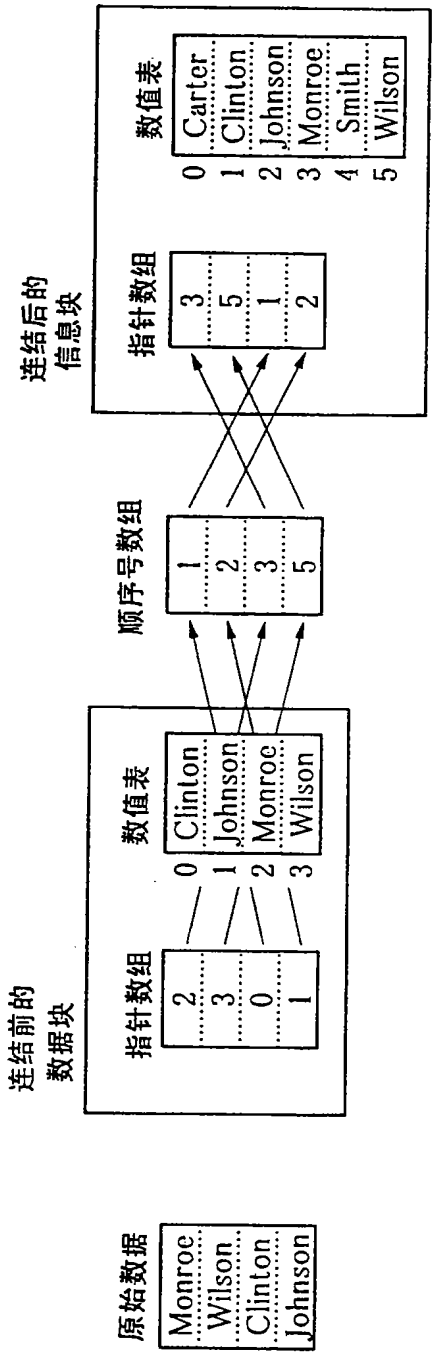

连结后的
信息块
指针数组
3
5
1
2
数值表
0 Carter
1 Clinton
2 Johnson
3 Monroe
4 Smith
5 Wilson
顺序号数组
1
2
3
5
连结前的
数据块
指针数组
2
3
0
1
数值表
0 Clinton
1 Johnson
2 Monroe
3 Wilson
原始数据
Monroe
Wilson
Clinton
Johnson

图 44A

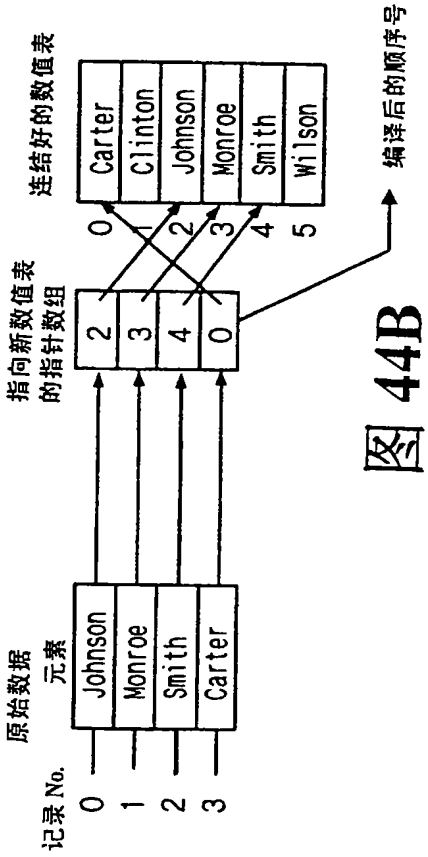

原始数据
元素
记录 No.
Johnson
Monroe
Smith
Carter
0
1
2
3
指向新数值表
的指针数组
2
3
4
0
连结好的数值表
Carter
Clinton
Johnson
Monroe
Smith
Wilson
0
1
2
3
4
5
编译后的顺序号

图 44B

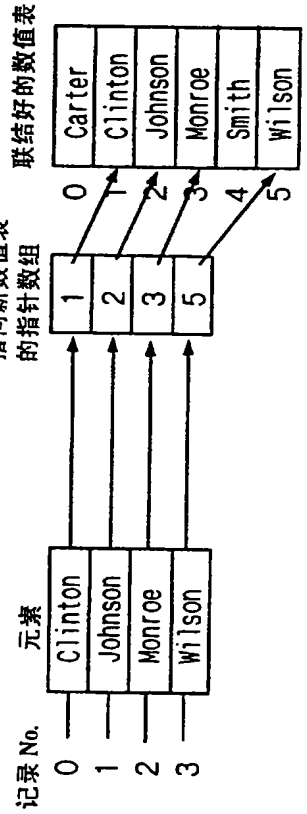

原始数据
元素
记录 No.
Clinton
Johnson
Monroe
Wilson
0
1
2
3
指向新数值表
的指针数组
1
2
3
5
联结好的数值表
Carter
Clinton
Johnson
Monroe
Smith
Wilson
0
1
2
3
4
5

图 45